



# **SAS<sup>®</sup> High-Performance Forecasting 2.3**

User's Guide

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2007. *SAS® High-Performance Forecasting 2.3: User's Guide*. Cary, NC: SAS Institute Inc.

### **SAS® High-Performance Forecasting 2.3: User's Guide**

Copyright © 2007, SAS Institute Inc., Cary, NC, USA

ISBN 978-1-59994-570-5

All rights reserved. Produced in the United States of America.

**For a hard-copy book:** No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**For a Web download or e-book:** Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

**U.S. Government Restricted Rights Notice:** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19, Commercial Computer Software-Restricted Rights (June 1987).

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

1st printing, December 2007

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at [support.sas.com/pubs](http://support.sas.com/pubs) or call 1-800-727-3228.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

# Contents

---

Acknowledgments . . . . .	i
What's New in SAS High-Performance Forecasting 2.3 . . . . .	iii
<b>I General Information</b>	<b>1</b>
Chapter 1. Introduction . . . . .	3
<b>II Procedure Reference</b>	<b>21</b>
Chapter 2. The HPFARIMASPEC Procedure . . . . .	23
Chapter 3. The HPFDIAGNOSE Procedure . . . . .	39
Chapter 4. The HPFENGINE Procedure . . . . .	103
Chapter 5. The HPFESMSPEC Procedure . . . . .	175
Chapter 6. The HPFEVENTS Procedure . . . . .	187
Chapter 7. The HPFEXMSPEC Procedure . . . . .	231
Chapter 8. The HPFIDMSPEC Procedure . . . . .	237
Chapter 9. The HPFRECONCILE Procedure . . . . .	249
Chapter 10. The HPFSELECT Procedure . . . . .	281
Chapter 11. The HPFUCMSPEC Procedure . . . . .	305
Chapter 12. The HPF Procedure . . . . .	327
<b>III Forecasting Details</b>	<b>379</b>
Chapter 13. Forecasting Process Summary . . . . .	381
Chapter 14. Forecasting Process Details . . . . .	411
Chapter 15. Using Forecasting Model Score Files and DATA Step Functions . . . . .	447
Chapter 16. Using User-Defined Models . . . . .	453
Chapter 17. Using External Forecasts . . . . .	463

**Subject Index** 467

**Syntax Index** 471

# Acknowledgments

## Contents

---

Credits . . . . .	i
Documentation . . . . .	i
Software . . . . .	i
Technical Support . . . . .	ii

---

---

## Credits

---

### Documentation

Editing	Ed Huddleston
Technical Review	Ming-Chun Chang, Bruce Elsheimer, Wilma S. Jackson, Jennifer 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
Testing	Ming-Chun Chang, Bruce Elsheimer, Jennifer Lee, Michael J. Leonard

---

## Technical Support

Members

Donna E. Woodward, Kevin Meyer

# What's New in SAS High-Performance Forecasting 2.3

## Contents

---

Overview . . . . .	iii
Details . . . . .	iv
High-Performance Forecasting Release Numbering Scheme . . . . .	iv
HPF Procedure . . . . .	iv
HPFDIAGNOSE Procedure . . . . .	iv
HPFENGINE Procedure . . . . .	v
HPFEVENTS Procedure . . . . .	v
HPFRECONCILE Procedure . . . . .	vi

---

---

## Overview

**NOTE:** This section describes the features of SAS High-Performance Forecasting that are new or enhanced since SAS 9.1.3.

SAS High-Performance Forecasting has a new release numbering scheme. SAS High-Performance Forecasting 2.3 provides new features while maintaining all the capabilities of SAS 9.1.3 High-Performance Forecasting software.

New features have been added to the following procedures:

- [HPF](#) procedure
- [HPFDIAGNOSE](#) procedure
- [HPFENGINE](#) procedure
- [HPFEVENTS](#) procedure
- [HPFRECONCILE](#) procedure

---

## Details

---

### High-Performance Forecasting Release Numbering Scheme

SAS High-Performance Forecasting has a new release numbering scheme. SAS High-Performance Forecasting 2.3 provides the same features and functionality as SAS 9.1.3 High-Performance Forecasting software and includes new features.

---

### HPF Procedure

New features related to the forecast model selection enable you to better monitor forecast generation.

The **HPF** procedure has the following new features:

- **OUTPROCINFO=** option specifies the output data set to contain the summary information of the processing done by HPF Procedure. This option is particularly useful for easy programmatic assessment of the status of the procedure's execution via a data set instead of looking at or parsing the SAS log.
- **statistics of fit** related to scaled and relative errors.

---

### HPFDIAGNOSE Procedure

Many new features related to the diagnostics enable you to better control model generation.

The **HPFDIAGNOSE** procedure has the following features:

- **DELAYEVENT=** option specifies the delay lag for the events. If the option is not specified, the delay lag for the events is set to zero by default.
- **DELAYINPUT=** option specifies the delay lag for the inputs. If the option is not specified, the delay lag for the inputs is chosen appropriately by the procedure.
- **ENTRYPCT=** option specifies a threshold to check the percentage increment of the criterion between two candidate models. The **ENTRYPCT=**value should be in the range (0,100); the default is **ENTRYPCT=0.1** (0.1%).
- **NODIAGNOSE** option specifies that the series is not to be diagnosed. If the **INSELECT-NAME=** option and **OUTEST=** option are specified, the existing model specification files are written to the **OUTEST** data set.



- **OUTPROCINFO=** option specifies the output data set to contain the summary information of the processing done by the HPFDIAGNOSE procedure. This option is particularly useful for easy programmatic assessment of the status of the procedure's execution via a data set instead of looking at or parsing the SAS log.
- **OUTLIER=(ENTRYPCT=)** option specifies a threshold to check the percentage increment of the criterion between two candidate models. The ENTRYPCT=value should be in the range (0,100); the default is ENTRYPCT=0.1 (0.1%). The value of the OUTLIER=(ENTRYPCT=) option overrides the value of the ENTRYPCT= option in the HPFDIAGNOSE statement.
- **REFINEPARMS=** option specifies that insignificant parameters of the final model are to be refined, identifies the factors to refine, and identifies the order of factors.
- **RETAINCHOOSE** option specifies that the PROC HPFSELECT CHOOSE= option is respected when re-diagnosing series.
- **POSITIVE** or **NEGATIVE** option is specified in the INPUT statement followed by the REQUIRED= option such as REQUIRED=YES(POSITIVE) and REQUIRED=MAYBE(NEGATIVE). When the REQUIRED=YES(POSITIVE) option is specified, if its coefficient is negative, then the input variable drops out from the model.

---

## HPFENGINE Procedure

New features related to the forecasting engine enable you to better monitor forecast generation.

The new **HPFENGINE** procedure has the following features:

- **OUTPROCINFO=** option specifies the output data set to contain the summary information of the processing done by the HPFENGINE procedure. This option is particularly useful for easy programmatic assessment of the status of the procedure's execution via a data set instead of looking at or parsing the SAS log.
- **statistics of fit** related to scaled and relative errors.

---

## HPFEVENTS Procedure

New predefined date keywords have been added to the **HPFEVENTS** procedure:

**Table 3** New Holiday Date Keywords and Definitions

Date Keyword	Definition
COLUMBUS	2nd Monday in October
MLK	3rd Monday in January
USPRESIDENTS	3rd Monday in February (since 1971)
VETERANS	November 11
VETERANSUSG	date observed by U.S. government for Monday-Friday schedule
VETERANSUSPS	date observed by U.S. government for Monday-Saturday schedule (U.S. Post Office)

---

## HPFRECONCILE Procedure

The following new options are available in the [HPFRECONCILE](#) procedure:

- **FORCECONSTRAINT** option specifies whether the user-specified constraints should be forced on the PREDICT variable in the OUTFOR= data set when the problem is infeasible because the constraints are incompatible with the aggregation constraint. The default is to leave the input unmodified.
- **OUTPROCINFO= SAS-data-set** specifies the output data set to contain the summary information of the processing done by PROC HPFRECONCILE. This option is particularly useful for easy programmatic assessment of the status of the procedure's execution via a data set instead of looking at or parsing the SAS log.
- **WEIGHTED** option specifies that the loss function for top-down reconciliation be weighted by the inverse of the variance of the input forecasts.
- **OUTINFEASIBLE= data set** specifies the output data set that contains summary information.
- **ALIGN=** option controls the alignment of SAS dates used to identify output observations. Internal processing uses aligned versions of the values of START= and END= options (if specified) and values of ID variable in input observations. The ALIGN= option accepts the following values: BEGIN, MIDDLE, and END. BEGIN is the default.

## **Part I**

# **General Information**



# Chapter 1

## Introduction

### Contents

---

Overview of SAS High-Performance Forecasting Software . . . . .	<b>3</b>
Uses of SAS High-Performance Forecasting Software . . . . .	5
Contents of SAS High-Performance Forecasting Software . . . . .	5
About This Book . . . . .	<b>8</b>
Chapter Organization . . . . .	8
Typographical Conventions . . . . .	9
Options Used in Examples . . . . .	9
Where to Turn for More Information . . . . .	<b>11</b>
Accessing the SAS High-Performance Forecasting Sample Library . . . . .	11
Online Help System . . . . .	11
SAS Institute Technical Support Services . . . . .	11
Related SAS Software . . . . .	<b>12</b>
Base SAS Software . . . . .	12
SAS/GRAPH Software . . . . .	14
SAS/STAT Software . . . . .	15
SAS/IML Software . . . . .	15
SAS/INSIGHT Software . . . . .	16
SAS/OR Software . . . . .	17
SAS/QC Software . . . . .	18
Other Statistical Tools . . . . .	18
References . . . . .	<b>19</b>

---

---

## 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	<p>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.</p> <p>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.</p> <p>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.</p>
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.
<i>oblique</i>	is used for user-supplied values for options in the syntax definitions. In the text, these values are written in <i>italic</i> .
helvetica	is used for the names of variables and data sets when they appear in the text.
<b>bold</b>	is used to refer to matrices and vectors, and to refer to commands (e.g., <b>end</b> or <b>cd</b> .)
<i>italic</i>	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 = pslepsz
          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 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 <a href="http://support.sas.com/documentation/onlinedoc">http://support.sas.com/documentation/onlinedoc</a> by selecting “Base SAS” from the Product-Specific Documentation list
TABULATE	for printing descriptive statistics in tabular format
TIMEPLOT	for plotting variables over time
TRANPOSE	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.



## **Part II**

# **Procedure Reference**





## Chapter 2

# The HPFARIMASPEC Procedure

### Contents

---

Overview . . . . .	23
Getting Started . . . . .	23
Syntax . . . . .	25
Functional Summary . . . . .	25
PROC HPFARIMASPEC Statement . . . . .	26
FORECAST Statement . . . . .	27
INPUT Statement . . . . .	29
ESTIMATE Statement . . . . .	31
Examples . . . . .	32
Example 2.1: Some Syntax Illustrations . . . . .	32
Example 2.2: How to Include ARIMA Models in a Model Selection List . . . . .	33
Example 2.3: A Generic Seasonal Model Specification Suitable for Different Season Lengths . . . . .	35
References . . . . .	37

---

---

## 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 4, “[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 6, “[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
<b>Model Repository Options</b>		
specify the model repository	PROC MASPEC	HPFARI- REPOSITORY=
specify the model specification name	PROC MASPEC	HPFARI- NAME=
specify the model specification label	PROC MASPEC	HPFARI- LABEL=
<b>Options for Specifying Symbolic Series Names</b>		
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=
<b>Options for Specifying the Model</b>		
specify the response series transformation	FORECAST	TRANSFORM=
specify the response series transformation options	FORECAST	TRANSOPT=
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=

Description	Statement	Option
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*

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=.

## 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=** *variable*

**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 2.3](#).

**P=** *order*

**P=** (*lag, ..., lag*) ... (*lag, ..., lag*)

**P=** (*lag, ..., lag*)<*s*<sub>1</sub>> ... (*lag, ..., lag*)<*s*<sub>*k*</sub>>

specifies the autoregressive part of the model. By default, no autoregressive parameters are fit.

$P=(l_1, l_2, \dots, l_k)$  defines a model with autoregressive parameters at the specified lags.  $P=order$  is equivalent to  $P=(1, 2, \dots, order)$ .

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 2.3](#).

**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  $\mu$  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.

**TRANSOPT= option**

specifies mean or median forecasts should be estimated. The following options are provided:

MEAN	Mean forecasts are estimated. This is the default.
------	--

MEDIAN          Median forecasts are estimated.

If no transformation is applied to the actual series using the TRANSFORM= option, the mean and median time series forecast values are identical.

---

## 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 )<sub>< s<sub>1</sub> ></sub> ... ( lag, ..., lag )<sub>< s<sub>k</sub> ></sub>**

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 )<sub>< s<sub>1</sub> ></sub> ... ( lag, ..., lag )<sub>< s<sub>k</sub> ></sub>**

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( <i>n</i> )	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 2.1: 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 *modell*, *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=promol
        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 2.2: 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 4, “[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 10, “[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 CO<sub>2</sub>. The goal is to forecast the output CO<sub>2</sub>, 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 @@;
    date = intnx( 'day', '01jan1950'd, _n_-1 );
    format date DATE.;
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
... more lines ...

```

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=m1;
run;

* make spec2;
proc hpfarimaspec repository=work.mycat name=m2;
    forecast symbol=y p=2;
    input symbol=x delay=3;
    estimate method=m1;
run;

* make spec3;
proc hpfarimaspec repository=work.mycat
    name=m3;
    forecast symbol=y p=2;
    estimate method=m1;
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 [Figure 2.2.1](#)).

```

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

```

**Output 2.2.1** Model Selection Based on the MAPE Criterion

The HPFENGINE Procedure		
Model Selection		
Criterion = MAPE		
Model	Statistic	Selected
M1	0.31478457	Yes
M2	0.50671996	No
M3	0.53295590	No
Model Selection Criterion = MAPE		
Model	Label	
M1	ARIMA: Y ~ P = 2 + INPUT: Lag(3) X NUM = 2 DEN = 1	
M2	ARIMA: Y ~ P = 2 + INPUT: Lag(3) X	
M3	ARIMA: Y ~ P = 2	

## Example 2.3: A Generic Seasonal Model Specification 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 [Figure 2.3.1](#) and [Figure 2.3.2](#), 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 2.3.1** Parameter Estimates for the Monthly Series

The HPFENGINE Procedure					
Parameter Estimates for GENERICAIRLINE Model					
Component	Parameter	Estimate	Standard Error	t Value	Approx 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 2.3.2 Parameter Estimates for the Quarterly Series

The HPFENGINE Procedure					
Parameter Estimates for GENERICAIRLINE Model					
Component	Parameter	Estimate	Standard Error	t Value	Approx 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 3

## The HPFDIAGNOSE Procedure

### Contents

---

Overview: HPFDIAGNOSE Procedure . . . . .	<b>40</b>
Getting Started: HPFDIAGNOSE Procedure . . . . .	<b>41</b>
Default Settings . . . . .	45
The Role of the IDM Statement . . . . .	46
Syntax: HPFDIAGNOSE Procedure . . . . .	<b>47</b>
Functional Summary . . . . .	47
PROC HPFDIAGNOSE Statement . . . . .	50
ADJUST Statement . . . . .	57
ARIMAX Statement . . . . .	58
BY Statement . . . . .	61
ESM Statement . . . . .	61
EVENT Statement . . . . .	61
FORECAST Statement . . . . .	62
ID Statement . . . . .	63
IDM Statement . . . . .	64
INPUT Statement . . . . .	65
TRANSFORM Statement . . . . .	66
TREND Statement . . . . .	67
UCM Statement . . . . .	68
Details: HPFDIAGNOSE Procedure . . . . .	<b>69</b>
Adjustment Operations . . . . .	69
Data Preparation . . . . .	70
Functional Transformation . . . . .	72
Stationarity Test . . . . .	73
ARMA Order Selection . . . . .	76
Transfer Functions in an ARIMAX Model . . . . .	77
Outliers . . . . .	79
Intermittent Demand Model . . . . .	80
Exponential Smoothing Model . . . . .	81
Unobserved Components Model . . . . .	82
Values of Status . . . . .	84
Holdout Sample . . . . .	85
EVENTS . . . . .	86
HPFENGINE . . . . .	88

Output Data Sets . . . . .	90
ODS Table Names . . . . .	94
Examples: HPFDIAGNOSE Procedure . . . . .	<b>95</b>
Example 3.1: Selection of Input Variables . . . . .	95
Example 3.2: Selection of Events and Input Variables . . . . .	96
Example 3.3: Intermittent Demand Series . . . . .	98
Example 3.4: Exponential Smoothing Model . . . . .	99
Example 3.5: Unobserved Components Model . . . . .	100
References . . . . .	<b>101</b>

---

## Overview: HPFDIAGNOSE Procedure

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)

PROC HPFDIAGNOSE may be abbreviated as PROC HPFDIAG.

---

## Getting Started: HPFDIAGNOSE Procedure

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 hpfdiagnose data=sashelp.air
                seasonality=12
                print=short;
    forecast air;
    transform;
    arimax;
run;
```

Figure 3.1 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)<sub>12</sub> 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.

Figure 3.1 ARIMAX Specification

The HPFDIAGNOSE Procedure											
ARIMA Model Specification											
Variable	Functional Transform	Constant	p	d	q	P	D	Q	Seasonality	Model Criterion	Statistic
AIR	LOG	NO	1	1	0	0	1	1	12	RMSE	10.8353
ARIMA Model Specification											
Variable Status											
AIR	OK										

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;
  transform;
  esm;
run;
```

Figure 3.2 shows the ESM model specification. The chosen model specification applied the log transformation and selected a multiplicative seasonal model with a trend component (WINTERS).

Figure 3.2 ESM Specification

The HPFDIAGNOSE Procedure					
Exponential Smoothing Model Specification					
Variable	Functional Transform	Selected Model	Component	Model Criterion	Statistic
AIR	LOG	WINTERS	LEVEL TREND SEASONAL	RMSE	10.6521

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;
  transform;
```

```

ucm;
run;

```

When the column `SELECTED=YES`, the component is significant. When the column `SELECTED=NO`, the component is insignificant in [Figure 3.3](#).

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 3.3](#) 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.

**Figure 3.3** Select Components

The HPFDIAGNOSE Procedure						
Unobserved Components Model (UCM) Specification						
Variable	Functional Transform	Component	Selected	Stochastic	Seasonality	Model 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				

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 datasets lib=sasuser mt=catalog nolist;
  delete hpfscor mycat;
run;

```

```

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

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

```

Figure 3.4 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.

**Figure 3.4** Model Selection from the HPFENGINE procedure

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		

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;
  transform;
  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 3.5 shows the three model specifications (DIAG2, DIAG3, DIAG4) found by the HPFDIAGNOSE procedure.

**Figure 3.5** Model Selection

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		

## Default Settings

The following example shows the HPFDIAGNOSE procedure with the default settings. The data sets AAA, BBB, and CCC are not specific data sets.

```
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=none;
  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: HPFDIAGNOSE Procedure

The HPFDIAGNOSE procedure uses the following statements:

```

PROC HPFDIAGNOSE options ;
  ADJUST variable = ( variable-list ) / options ;
  ARIMAX options ;
  BY variables ;
  ESM option ;
  EVENT event-names ;
  FORECAST variables ;
  ID variable INTERVAL=interval options ;
  IDM options ;
  INPUT variables ;
  TRANSFORM options ;
  TREND options ;
  UCM options ;

```

---

## Functional Summary

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

Description	Statement	Option
<b>Statements</b>		
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	

---

Description	Statement	Option
<b>Model Repository Options</b>		
specifies the model repository	HPFDIAGNOSE	REPOSITORY=
specifies the base name for model specification files or model selection list files	HPFDIAGNOSE	BASENAME=
<b>Data Set Options</b>		
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=
specifies the output data set that contains the information in the SAS log	HPFDIAGNOSE	OUTPROCINFO=
<b>Accumulation Options</b>		
specifies length of seasonal cycle	HPFDIAGNOSE	SEASONALITY=
specifies accumulation frequency	ID	INTERVAL=
specifies interval alignment	ID	ALIGN=
specifies starting time ID value	ID	START=
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=
<b>Transformation Test Options</b>		
specifies the AR order for the log transformation test	TRANSFORM	P=
specifies the type of the functional transformation	TRANSFORM	TYPE=

Description	Statement	Option
specifies the method of the forecasts of the transformed series	TRANSFORM	TRANSOPT=
<b>Trend Test Options</b>		
specifies the simple differencing	TREND	DIFF=
specifies the seasonal differencing	TREND	SDIFF=
specifies the AR order for the augmented unit root test	TREND	P=
<b>ARIMAX Model Options</b>		
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=
specifies the outlier detection	ARIMAX	OUTLIER=
specifies the identification order of the components	ARIMAX	IDENTIFYORDER=
<b>Unobserved Components Model Option</b>		
specifies the components to test for inclusion in the UCM model	UCM	COMPONENT=
<b>Exponential Smoothing Model Option</b>		
specifies the method of the ESM model	ESM	METHOD=
<b>Significance Level Option</b>		
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=
<b>Event Variable Control Option</b>		
specifies the maximum number of the events to be selected	HPFDIAGNOSE	SELECTEVENT=
specifies the required option of the event	EVENT	REQUIRED=

Description	Statement	Option
<b>Input Variable Control Options</b>		
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=
<b>Model Selection Options</b>		
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=
specifies the threshold of forward selection of inputs and events	HPFDIAGNOSE	ENTRYPCT=
<b>Printing Options</b>		
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=
<b>Data Prefilter Option</b>		
specifies handling missing and extreme values prior to diagnostic tests	HPFDIAGNOSE	PREFILTER=

## PROC HPFDIAGNOSE Statement

**PROC HPFDIAGNOSE** *options* ;

**PROC HPFDIAG** *options* ;

The following options can be used in the PROC HPFDIAGNOSE or HPFDIAG 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  $\text{BACK}=n$  and the number of observation is  $T$ , then the first  $T - n$  observations are used to diagnose a series. The default is  $\text{BACK}=0$ .

**BASENAME=** *SAS-name*

prefixes the model specification file name and/or the model selection list file name. If the  $\text{BASENAME}=\text{MYSPEC}$ , then the model specification files and/or the model selection list files are named  $\text{MYSPEC0}, \dots, \text{MYSPEC999999999}$ . The default SAS-name starts with  $\text{DIAG}$ , such as  $\text{DIAG0}, \dots, \text{DIAG999999999}$ . The model specification files and/or the model selection list files are stored in the model repository defined by the  $\text{REPOSITORY}=\text{option}$ .

**CRITERION=***option*

specifies the model selection criterion to select the best model. This option would often be used in conjunction with the  $\text{HOLDOUT}=\text{}$  and  $\text{HOLDOUTPCT}=\text{}$  options. The default is  $\text{CRITERION}=\text{RMSE}$ . 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
MAPES	Mean Absolute Error Percent of Standard Deviation
MDAPES	Median Absolute Error Percent of Standard Deviation
GMAPES	Geometric Mean Absolute Error Percent of Standard Deviation
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
MASE	Mean Absolute Scaled Error
RSQUARE	R-Square
ADJRSQ	Adjusted R-Square
AADJRSQ	Amemiya's Adjusted R-Square
RWRSQ	Random Walk R-Square
AIC	Akaike Information Criterion
AICC	Akaike Information Corrected 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.

**DELAYEVENT=** *number*

specifies the delay lag for the events. If the option is not specified, the delay lag for the events is set to zero by default.

**DELAYINPUT=** *number*

specifies the delay lag for the inputs. If the option is not specified, the delay lag for the inputs is appropriately chosen by the procedure.

**ENTRYPCT=** *number*

specifies a threshold to check the percentage increment of the criterion between two candidate models. The ENTRYPCT=value should be in (0,100); the default is ENTRYPCT=0.1.

**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)
```

**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.

**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.

**INEST= SAS data set**

contains information that maps forecast variables to models or selection lists, and data set variables to model variables.

**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 6, “[The HPFEVENTS Procedure](#).”

**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.

**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 nonmissing observations than  $number$ . The value of  $number$  must be greater than or equal to 1. The default is  $number = 1$ .

**NODIAGNOSE**

specifies that the series is not diagnosed. If the INSELECTNAME= option and OUTEST= option are specified, the existing model specification files are written to the OUTEST data set.

**NOINESTOPTS**

specifies that the selection lists referred by the INEST= option are not used in the diagnosed version the series is not diagnosed.



**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.

**OUTPROCINFO= SAS-data-set**

names the output data set to contain the summary information of the processing done by PROC HPFDIAGNOSE. It is particularly useful for easy programmatic assessment of the status of the procedure's execution via a data set instead of looking at or parsing the SAS log.

**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.

**RETAINCHOOSE**

specifies that the CHOOSE= option in the HPFSELECT procedure is respected when re-diagnosing series.

**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).
<i>number</i>	selects the best <i>number</i> 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).
<i>number</i>	selects the best <i>number</i> 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.

**SELECTBASE= SAS-name**

prefixes the model selection list file name. If the SELECTBASE=MYSELECT, then the model selection list files are named MYSELECT0, . . . . The default SAS-name starts with DIAG, such as DIAG0, . . . . The model selection list files are stored in the model repository defined by the REPOSITORY= option.

**SPECBASE= SAS-name**

prefixes the model specification file name. If the SPECBASE=MYSPEC, then the model specification files are named MYSPEC0, . . . . The default SAS-name starts with DIAG, such as DIAG0, . . . . The model specification files are stored in the model repository defined by the REPOSITORY= option.

**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.

---

## 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](#)” on page 69 section.

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.



**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.

**ENTRYPCT=number** specifies a threshold to check the percentage increment of the criterion between two candidate models. The **ENTRYPCT=value** should be in (0,100); the default is **ENTRYPCT=0.1**. The **ENTRYPCT=** option overrides the value of the **ENTRYPCT=** 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.

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

**REFINEPARMS=( options )**

specifies to do refining insignificant parameters of the final model, identifying the factors to refine, and identifying the order of factors.

**SIGLEVEL=** specifies the cutoff value for all refining insignificant parameters. The **SIGLEVEL=value** should be between (0,1) and **SIGLEVEL=0.4** is the default.

**FACTOR=ALL** refines the parameters for all factors. This option is the default.

**FACTOR=ARMA** refines the parameters for ARMA factor.

**FACTOR=EVENT** refines the parameters for EVENT factor.

**FACTOR=INPUT** refines the parameters for INPUT factor.

Using parentheses, more than one option can be specified. For example, the option **FACTOR=( ARMA EVENT )** refines the parameters for ARMA and EVENT.

The **FIRST** and **SECOND** options take one of the factors ARMA, EVENT, and INPUT.

**FIRST=** specifies the factor which refines first.

**SECOND=** specifies the factor which refines second.

The default order of refining is an order of ARMA, EVENT, INPUT.

---

## 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.

---

## 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).

---

## 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 6, “[The HPFEVENTS Procedure](#),”

The following option can be used in the EVENT statement:

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

YES	specifies that the events be included in the model as long as the model does not fail to be diagnosed.
MAYBE	specifies that the events be included in the model as long as the parameters of events are significant.
NO	specifies that the events be included in the model as long as the parameters of events are significant and the increment of the value of criterion exceeds a threshold. 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](#)” on page 63 section for more details.

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

See the SETMISSING= option in the “[ID Statement](#)” on page 63 section for more details.

**TRIMMISS=***option*

See the TRIMMISS= option in the “[ID Statement](#)” on page 63 section for more details.

**ZEROMISS=***option*

See the ZEROMISS= option in the “[ID Statement](#)” on page 63 section 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 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” on page 123 section 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 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.

---

## 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.

---

## 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*

YES	specifies that the input variables be included in the model as long as the model does not fail to be diagnosed.
MAYBE	specifies that the input variables be included in the model as long as their parameters are significant.
NO	specifies that the input variables be included in the model as long as their parameters are significant and the increment of the value of criterion exceeds a threshold. The default is REQUIRED=NO.

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.

Either the POSITIVE or NEGATIVE option with parentheses can follow the REQUIRED= option. For examples REQUIRED=YES(POSTIVE) and REQUIRED=MAYBE(NEGATIVE). When the REQUIRED=YES(POSTIVE) option is specified, if its coefficient is negative, then the input variable drops out from to the model.

**ACCUMULATE=option**

See the ACCUMULATE= option in the “ID Statement” on page 63 section for more details.

**SETMISSING=option | number**

See the SETMISSING= option in the “ID Statement” on page 63 section for more details.

**TRIMMISS=option**

See the TRIMMISS= option in the “ID Statement” on page 63 section for more details.

**ZEROMISS=option**

See the ZEROMISS= option in the “ID Statement” on page 63 section 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 specified but the TYPE= option is not specified, then the TYPE=AUTO is the default.
LOG	Logarithmic transformation.
NONE	No transformation is applied. If the TRANSFORM statement is not specified, the TYPE=NONE option is the default.
SQRT	Square-root transformation.
LOGISTIC	Logistic transformation.
BOXCOX( <i>value</i> )	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.
<i>number</i>	Specifies the simple differencing order. The option <i>number</i> =1 means $(1 - B)y_t$ and <i>number</i> =2 means $(1 - B)^2 y_t$ .
( <b>0</b> : <i>number</i> )	Specifies the range of simple differencing order for testing. The option <i>number</i> 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  $(1 - B^s)^2 y_t$  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.

## 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 <i>autoreg</i> component is significant in the model.
CYCLE	tests if two <i>cycle</i> 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 <i>dependent lag</i> component is significant in the model. Only the order 1 is included.
IRREGULAR	tests if an <i>irregular</i> component is significant in the model.
LEVEL	tests if a <i>level</i> component is significant in the model.
SEASON	tests if a <i>season</i> 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.

**REFINEPARMS= ( SIGLEVEL= | FACTOR=(ALL|EVENT|INPUT) | FIRST=EVENT|INPUT )**

specifies to do refining insignificant parameters of the final model, identifying the factors to refine, and identifying the order of factors.

**SIGLEVEL=** specifies the cutoff value for all refining insignificant parameters. The SIGLEVEL=value should be between (0,1) and SIGLEVEL=0.4 is the default.

**FACTOR=ALL** refines the parameters for all factors. This option is the default.

**FACTOR=EVENT** refines the parameters for EVENT factor.

**FACTOR=INPUT** refines the parameters for INPUT factor.

Using parentheses, more than one option can be specified. For example, the option FACTOR=( EVENT INPUT ) refines the parameters for ARMA and EVENT.

**FIRST=** specifies the factor which refines first.

The default order of refining is an order of EVENT, INPUT.

## Details: HPFDIAGNOSE Procedure

### 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  $i = 1, \dots, M$  are the  $M$  adjustment series, then the adjusted dependent series,  $w_t$ , is

$$\begin{aligned} w_t^1 &= op_1(y_t, z_{1,t}) \\ w_t^i &= op_i(w_t^{i-1}, z_{i,t}) \text{ for } 1 < i \leq M \\ w_t &= w_t^M \end{aligned}$$

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

---

## 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  $3 \times \text{STDDEV}$  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;
run;
```

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

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

In [Figure 3.6](#), 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 3.1](#). The three extreme values mislead the model diagnostic tests.



**Figure 3.6** ARIMAX Model Specification without Outliers

```

The HPFDIAGNOSE Procedure

ARIMA Model Specification

      Functional
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

```

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

```

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

```

In [Figure 3.7](#), the ARIMA(1, 1, 0)(0, 1, 0)<sub>12</sub> model is diagnosed for the time series. The required seasonal differencing is detected.

**Figure 3.7** ARIMAX Model Specification without Outliers

```

The HPFDIAGNOSE Procedure

ARIMA Model Specification

      Functional
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 3.8](#) shows that the three extreme values are detected as outliers.

**Figure 3.8** Outlier Information

ARIMA Outlier Selection					
Variable	Type	Obs	Time	Chi-Square	Approx Pr > ChiSq
AIR	AO	100	APR1957	1875.29	<.0001
	AO	30	JUN1951	1820.42	<.0001
	AO	50	FEB1953	1836.58	<.0001

After the three extreme values are included in the ARIMAX model, [Figure 3.8](#) shows that the statistic of the model selection criterion dramatically drops from 85.7080 to 11.5489.

**Figure 3.9** ARIMAX Model Specification with Outliers

ARIMA Model Specification After Adjusting for Outliers											
Variable	Functional Transform	Constant	p	d	q	P	D	Q	Seasonality	Outlier	Model 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

## 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;
  transform;
  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 3.10 states that the airline data requires a log transformation.

**Figure 3.10** Log Transformation Test

The HPFDIAGNOSE Procedure	
Functional Transformation Test	
Variable	Functional Transform
AIR	LOG

---

## 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;
  transform;
  arimax;
run;
```

```
proc hpfdiag data=sashelp.air print=all;
  id date interval=month;
  forecast air;
  transform;
  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 3.11 and Figure 3.12 show that the series needs simple differencing because the null hypothesis test probability is greater than SIGLEVEL=0.05.

**Figure 3.11** Dickey-Fuller Unit Root Test

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 3.12** Summary of Dickey-Fuller Unit Root Test

Dickey-Fuller Unit Root Test Summary				
Variable	Seasonality	Zero Mean	Mean	Trend
AIR	1	YES	YES	NO

### 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 3.13 and Figure 3.14 show that the series needs seasonal differencing because the null hypothesis test probability is greater than SIGLEVEL=0.05.

**Figure 3.13** Seasonal Dickey-Fuller Unit Root Test

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 3.14 *continued*

Figure 3.14 Summary of Seasonal Dickey-Fuller Unit Root Test

Seasonal Dickey-Fuller Unit Root Test Summary				
Variable	Seasonality	Zero Mean	Mean	Trend
AIR	12	YES	YES	

## 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 3.15 and Figure 3.16 show that the series needs both simple and seasonal differencing because the null hypothesis test probability is greater than  $SIGLEVEL=0.05$ .

Figure 3.15 Joint Unit Root Test

Joint Unit Root Test					
Type	F Value	Critical Values			Approx Pr > F
		90%	95%	99%	
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 3.16 Summary of Joint Unit Root Test

Joint Unit Root Test Summary				
Variable	Seasonality	Zero Mean	Mean	Trend
AIR	1, 12	NO	YES	

## 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;
  transform;
  arimax;
run;

proc hpfdiag data=sashelp.air print=all;
  id date interval=month;
  forecast air;
  transform;
  arimax method=minic p=(0:5) q=(0:5) criterion=sbc;
run;
```

Figure 3.17 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.

**Figure 3.17** Minimum Information Criterion

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

### 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 3.18 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.

**Figure 3.18** ARIMAX Specification

ARIMA Model Specification											
Variable	Functional Transform	Constant	p	d	q	P	D	Q	Seasonality	Model Criterion	Statistic
AIR	LOG	NO	1	1	0	0	1	1	12	RMSE	10.8353
ARIMA Model Specification											
Variable Status											
AIR	OK										

## 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 TESTIN-PUT=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  $x_{it}$  ( $e_{it}^x$ ).
2. Prewhiten  $y_t$  using this model and get the residual of  $y_t$  ( $e_{it}^y$ ).
3. Compute the cross-correlations between  $e_{it}^x$  and  $e_{it}^y$  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;
  transform;
  arimax;
run;
```

The ARIMA Input Selection Table shown in [Figure 3.19](#) 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.



Figure 3.19 ARIMA Input Selection

The HPFDIAGNOSE Procedure							
ARIMA Input Selection							
Input Variable	Selected	Functional Transform	d	Delay	Numerator	Denominator	Status
CCIUTC	NO	NONE	2	2	1	1	Not Improved
EEC	NO	NONE	2	2	0	0	Not Improved
EEGP	YES	NONE	2	8	1	0	OK
EXVUS	NO						Not Causal
FM1	NO						Not Causal
FM1D82	NO						Not Causal

## 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;

... more lines ...
```

The next two SAS programs result in the same outlier analysis.

```
proc hpfdiag data=hardware print=short;
  id date interval=month;
  forecast hardware;
  transform;
  arimax;
```

```

run;

proc hpfdiag data=hardware print=short;
  id date interval=month;
  forecast hardware;
  transform;
  arimax outlier=(detect=maybe maxnum=2 maxpct=2 siglevel=0.01);
run;

```

Figure 3.20 shows that the two level shifts (LS) occurred at the 96th (DEC1974) and 99th (MAR1975) observations.

**Figure 3.20** Outlier Information

The HPFDIAGNOSE Procedure					
ARIMA Outlier Selection					
Variable	Type	Obs	Time	Chi-Square	Approx Pr > ChiSq
hardware	LS	99	MAR1975	25.73	<.0001
	LS	96	DEC1974	29.64	<.0001

Figure 3.21 shows the ARIMA model specification with two outliers included in the model.

**Figure 3.21** ARIMAX Specification

ARIMA Model Specification After Adjusting for Outliers										
Variable	Functional Transform	Constant	p	d	q	P	D	Q	Seasonality	Model 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							

## 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;
  transform;
  idm intermittent=2.5 base=0;
run;

```

Output 3.22 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.

**Figure 3.22** Intermittent Demand Model Specification

The HPFDIAGNOSE Procedure						
Intermittent Demand Model Specification						
Variable	Demand Model	Functional Transform	Selected Model	Component	Model Criterion	Statistic
hubcaps	INTERVAL	NONE	DOUBLE	LEVEL	RMSE	0.8288
	SIZE	LOG	SIMPLE	LEVEL		
	AVERAGE	LOG	SIMPLE	LEVEL		

## 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;
  transform;
  esm;
run;

```

The ESM model specification in Figure 3.23 states that the damp-trend exponential smoothed model was automatically selected.

**Figure 3.23** ESM Specification

The HPFDIAGNOSE Procedure					
Exponential Smoothing Model Specification					
Variable	Functional Transform	Selected Model	Component	Model Criterion	Statistic
GNP	NONE	DAMPTREND	LEVEL TREND DAMP	RMSE	22.0750

## 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;

... more lines ...
```

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;
run;
```

The following SAS code 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;
  transform;
  ucm;
run;
```

Figure 3.24 shows the three significant inputs chosen.

**Figure 3.24** UCM Input Selection

The HPFDIAGNOSE Procedure				
UCM Input Selection				
Input Variable	Selected	Functional Transform	d	Delay
Shift1899	YES	NONE	0	0
Event1913	YES	NONE	0	0

Figure 3.25 shows the UCM model specification for the Nile data. The data has a significant cycle, level components, and the two inputs.

**Figure 3.25** UCM Specification

Unobserved Components Model (UCM) Specification							
Variable	Functional 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

The following example has the same results as [Figure 3.24](#). 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;
  transform;
  input Shift1899 Event1913 Event1877;
  ucm component=(level cycle);
run;
```

---

## Values of Status

The meaning of the different values of Status in the output is summarized as follows:

OK	Model fits successfully.
All Missing Obs	All series values are missing.
All Same Values	All series values are identical.
Collinearity	Multi-collinearity between input series.
Not Causal	A delay parameter is negative.
Insignificant	Some of the parameters are insignificant.

X Model Failed	Prewhitening model of input series failed.
Y Model Failed	$g(y)=f(x)$ model failed.
Not Improved	Input series does not improve benchmarking model.
Not One of Best	When the <code>SELECTINPUT=<math>n</math></code> and <code>SELECTEVENT=<math>n</math></code> are specified, it is not one of the best first $n$ inputs and events.
Unstable Model	Model is unstable.
May Not Converge	Model may not converge.
Small Variance	The series values have very small variance.
Singularity	Model is singular.
Modeling Error	A model cannot be fitted to the data.
Extreme Value	The series values are extremely large.
Error Transform	The data transformation failed.
Lack of Memory	The memory is insufficient.
Not Enough Data	The number of observations is insufficient.
Bad Arguments	The arguments are incorrect.
Non Positive Obs	The series values are either zero or negative.

The following are the messages for the intermittent model:

No Demand	There is no recorded demand.
Not Fitted Model	Model cannot be fitted to the data.
Not Selected	Model cannot be selected.
Not Predicted	Model cannot be predicted.
Not Initialized	Model cannot be initialized.
Negative Demand	The demand size is negative.

---

## Holdout Sample

A holdout sample is useful to find models that have better out-of-sample forecasts. If the `HOLD-OUT=` 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;
  transform;
  arimax;
run;
```

The ARIMA model specification in Figure 3.26 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.

**Figure 3.26** Use HOLDOUT Option

The HPFDIAGNOSE Procedure											
ARIMA Model Specification											
Variable	Functional Transform	Constant	p	d	q	P	D	Q	Seasonality	Model 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

## 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;
  transform;
  ucm component=(level cycle);
run;

```

Figure 3.27 shows the three significant events chosen.

**Figure 3.27** UCM Event Selection

The HPFDIAGNOSE Procedure	
UCM Event Selection	
Event Name	Selected
SHIFT1899	YES
EVENT1913	YES

Figure 3.28 shows the UCM model specification for the Nile data. The data has the significant cycle, level components, and the two events.

**Figure 3.28** UCM Specification

Unobserved Components Model (UCM) Specification							
Variable	Functional 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							

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;
  transform;
  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.

```
proc datasets lib=sasuser mt=catalog nolist;
  delete hpfscor mymodel;
run;
```

Create a diagnosed model specification.

```
proc hpfdiag data=sashelp.air outest=est
  modelrepository=sasuser.mymodel;
  id date interval=month;
  forecast air;
  transform;
  arimax;
run;
```

Create an ARIMA(0, 1, 1)(0, 1, 1)<sub>s</sub> 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 3.29 shows the DIAG0 and BENCHModel model specifications. The DIAG0.XML is cre-

ated by the HPFDIAGNOSE procedure and the BENCHModel is created by the HPFARIMASPEC procedure.

**Figure 3.29** Model Selection from the HPFENGINE 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		

```
proc datasets lib=sasuser mt=catalog nolist;
  delete hpfscor mymodel;
run;
```

## Output Data Sets

The HPFDIAGNOSE procedure can create the OUTEST=, OUTOUTLIER=, and OUTPROCINFO= data sets. In general, if diagnostics for a particular time series fail, 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.

### OUTEST= Data Set

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.

<code>_SELECT_</code>	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.
<code>_MODEL_</code>	Not applicable in the HPFDIAGNOSE procedure.
<code>_SCORE_</code>	Not applicable in the HPFDIAGNOSE procedure.
<code>_MODELVAR_</code>	Model symbol.
<code>_DSVAR_</code>	Data set variable name.
<code>_VARTYPE_</code>	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 datasets lib=sasuser mt=catalog nolist;
  delete hpfscor mymodel;
run;

proc hpfdiag data=sashelp.air outest=est_air
  modelrepository=sasuser.mymodel
  basename=airSpec;
  id date interval=month;
  forecast air;
  transform;
  arimax;
run;
  proc print data=est_air;
run;
```

Figure 3.30 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.

**Figure 3.30** OUTEST1

Obs	NAME	SELECT	MODEL	SCORE	MODELVAR	DSVAR	VARTYPE	STATUS
1	AIR	AIRSPEC1			Y	AIR	DEPENDENT	0

```
proc datasets lib=sasuser mt=catalog nolist;
  delete hpfscor mymodel;
run;
```

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 datasets lib=sasuser mt=catalog nolist;
  delete hpfscor myGNP;
run;

proc hpfdiag data=sashelp.gnp outest=est_gnp
  modelrepository=sasuser.myGNP
  basename=gnpSpec;
  id date interval=qtr;
  forecast consump invest;
  transform;
  esm;
  arimax;
run;

proc print data=est_gnp;
run;
```

Figure 3.31 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.

**Figure 3.31** OUTEST2

Obs	_NAME_	_SELECT_	_MODEL_	_SCORE_	_MODELVAR_	_DSVAR_	_VARTYPE_	_STATUS_
1	CONSUMP	GNPSPEC2			Y	CONSUMP	DEPENDENT	0
2	INVEST	GNPSPEC5			Y	INVEST	DEPENDENT	0

```
proc datasets lib=sasuser mt=catalog nolist;
  delete hpfscor myGNP;
run;
```

## OUTOUTLIER= Data Set

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* .

<code>_DIRECTION_</code>	Contains a direction, either UP for a positive effect or DOWN for a negative effect.
<code>_OBS_</code>	Contains the number of the observation where the outlier happens.
<code>_SASDATE_</code>	Contains the SAS date when the outlier happens.
<code>_EVENT_</code>	Contains the outlier's event name.
<code>_ESTIMATE_</code>	Contains the coefficient estimate.
<code>_CHISQ_</code>	Contains the chi-square statistic of the coefficient estimate.
<code>_PVALUE_</code>	Contains the $p$ -value of the coefficient estimate.

The following SAS code and [Figure 3.32](#) show the `OUTOUTLIER=` data set that contains information associated with the output in [Figure 3.20](#).

```
proc hpfdiag data=hardware outoutlier=outl;
  id date interval=month;
  forecast hardware;
  transform;
  arimax;
run;

proc print data=outl;
run;
```

**Figure 3.32** OUTOUTLIER Data Set

			D				E			P
			I		S		S			V
			R		A		T			A
			E		S		I		C	L
			C		D		M		H	U
	N	T	T	O	A		A		I	E
	A	Y	I	B	T		T		S	
O	M	P	O	S	E		E		Q	
b	E	E	N	S						
s										
1	hardware	LS	DOWN	99	01MAR75	LS01MAR1975D	-125.695	25.7273	.000000393	
2		LS	DOWN	96	01DEC74	LS01DEC1974D	-115.714	29.6352	.000000052	

## OUTPROCINFO= Data Set

The `OUTPROCINFO=` data set contains the following variables:

<code>_SOURCE_</code>	The source procedure that produces this data set
<code>_STAGE_</code>	Stage of the procedure execution for which the summary variable is reported
<code>_NAME_</code>	Name of the summary variable
<code>_LABEL_</code>	Description of the summary variable

`_VALUE_` Value of the summary variable

## 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 3.2](#):

**Table 3.2** ODS Tables Produced Using the HPFDIAGNOSE Procedure

ODS Table Name	Description	Statement	Option
<b>ODS Tables Created by the PRINT=SHORT Option</b>			
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		
<b>Additional ODS Tables Created by the PRINT=LONG Option</b>			
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
<b>Additional ODS Tables Created by the PRINT=ALL Option</b>			
DFTest	Dickey-Fuller Unit Root Test	TREND	DIFF
ESACF	Extended Sample Autocorrelation Function	ARIMAX	ESACF
ESACFPValues	<i>P</i> -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	<i>P</i> -values of SCAN	ARIMAX	SCAN
SeasonDFTest	Seasonal Dickey-Fuller Unit Root Test	TREND	SDIFF



## Examples: HPFDIAGNOSE Procedure

### Example 3.1: 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 hpfdiagnose data=sashelp.citimon(obs=141)
  testinput=both selectinput=all print=all;
  forecast conb;
  input cciutc eec eegp exvus fm1 fml82;
  transform;
  arimax;
run;
```

Output 3.1.1 shows that the ARIMA (0, 2, 1) model is diagnosed for the variable (CONB) to be forecast.

#### Output 3.1.1 ARIMAX Specification before Input Selection

The HPFDIAGNOSE Procedure							
ARIMA Model Specification							
Variable	Functional Transform	Constant	p	d	q	Criterion	Statistic Status
CONB	NONE	NO	0	2	1	RMSE	2318.33 OK

Output 3.1.2 shows that one input variable (EEGP ) is selected. The input variable needs a simple differencing.

#### Output 3.1.2 ARIMA Input Selection

ARIMA Input Selection						
Input Variable	Selected	Functional Transform	d	Delay Numerator	Denominator	Status
CCIUTC	YES	NONE	1	4	0	0 OK
EEC	NO					Not Causal
EEGP	NO	NONE	1	9	0	1 Unstable Model
EXVUS	NO					Not Causal
FM1	NO					Not Causal
FML82	NO					Not Causal

Output 3.1.3 shows that the RMSE model selection criterion with inputs is smaller than the model selection criterion without inputs and outliers.

**Output 3.1.3** ARIMAX Specification after Input Selection

ARIMA Model Specification After Adjusting for Inputs and Outliers									
Variable	Functional Transform	Constant	p	d	q	Outlier	Input	Model Criterion	Statistic
CONB	NONE	NO	0	2	1	2	1	RMSE	2017.65
ARIMA Model Specification After Adjusting for Inputs and Outliers									
		Variable	Status						
		CONB	OK						

## Example 3.2: 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;
  transform;
  arimax outlier=(detect=no);
run;
```

Output 3.2.1 shows the seasonal ARIMA  $(0, 2, 1)(2, 0, 0)_4$  model diagnosed for the variable (GNP) to be forecast.

**Output 3.2.1** ARIMAX Specification before Event Input Selection

```

The HPFDIAGNOSE Procedure

ARIMA Model Specification

      Functional
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 3.2.2 shows that the SHOCK and SHIFT events are significant.

**Output 3.2.2** ARIMA Event Selection

```

ARIMA Event Selection

      Event
      Name      Selected      d      D      Status
SHOCK      YES      2      0      OK
SHIFT      YES      2      0      OK
    
```

Output 3.2.3 shows that the input variable, CONSUMP, is selected in the model.

**Output 3.2.3** ARIMA Input Selection

```

ARIMA Input Selection

Input
Variable      Selected      Functional
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 Model
EXPORTS      Unstable Model
GOVT      Unstable Model
    
```

Output 3.2.4 shows that the RMSE model selection criterion with the events and input is smaller than that without the events and input.

#### Output 3.2.4 ARIMAX Specification after Event Input Selection

ARIMA Model Specification After Adjusting for Events and Inputs												
Variable	Functional Transform	Constant	p	d	q	P	D	Q	Seasonality	Input	Event	Model 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 3.3: 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;
  transform;
run;

```

Output 3.3.1 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 3.3.1** Intermittent Demand Model Specification

The HPFDIAGNOSE Procedure						
Intermittent Demand Model Specification						
Variable	Demand Model	Functional Transform	Selected Model	Component	Model Criterion	Statistic
tires	INTERVAL	NONE	DAMPTREND	LEVEL TREND DAMP	RMSE	0.8754
	SIZE	LOG	LINEAR	LEVEL TREND		
	AVERAGE	NONE	LINEAR	LEVEL TREND		0.6125

**Example 3.4: 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;
  transform;
  esm;
run;

```

Output 3.4.1 shows that the variable (INV ) to be forecast diagnosed the damped-trend exponential smoothing model.

**Output 3.4.1** Exponential Smoothing Model Specification

The HPFDIAGNOSE Procedure					
Exponential Smoothing Model Specification					
Variable	Functional Transform	Selected Model	Component	Model Criterion	Statistic
inv	NONE	DAMPTREND	LEVEL TREND DAMP	RMSE	26.2492

## Example 3.5: 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;

... more lines ...

proc hpfdiag data=ozone print=all;
  id date interval=month;
  forecast ozone;
  input x1 summer winter;
  transform;
  ucm;
run;

```

Output 3.5.1 shows that the input SUMMER is selected in the model.

### Output 3.5.1 UCM Input Selection

The HPFDIAGNOSE Procedure						
UCM Input Selection						
Input Variable	Selected	Functional Transform	d	D	Delay	Status
x1	NO	NONE	0	0	11	Insignificant
summer	YES	NONE	0	0	6	OK
winter	NO					Not Causal

Output 3.5.2 shows that the variable to be forecast is explained by the irregular, level and season components, and an input.

**Output 3.5.2** Unobserved Components Model Specification

Unobserved Components Model (UCM) Specification						
Variable	Functional Transform	Component	Selected	Stochastic	Seasonality	Model 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 4

## The HPFENGINE Procedure

### Contents

---

Overview: HPFENGINE Procedure . . . . .	<b>104</b>
Getting Started: HPFENGINE Procedure . . . . .	<b>105</b>
Syntax: HPFENGINE Procedure . . . . .	<b>107</b>
Functional Summary . . . . .	108
PROC HPFENGINE Statement . . . . .	110
ADJUST Statement . . . . .	119
BY Statement . . . . .	120
CONTROL Statement . . . . .	120
EXTERNAL Statement . . . . .	122
FORECAST Statement . . . . .	122
ID Statement . . . . .	123
INPUT Statement . . . . .	127
SCORE Statement . . . . .	128
STOCHASTIC Statement . . . . .	128
Details: HPFENGINE Procedure . . . . .	<b>130</b>
Accumulation . . . . .	131
Missing Value Interpretation . . . . .	132
Adjustment Operations . . . . .	133
Diagnostic Tests . . . . .	135
Model Selection . . . . .	135
Transformations . . . . .	136
Parameter Estimation . . . . .	136
Missing Value Modeling Issues . . . . .	136
Forecasting . . . . .	137
Inverse Transformations . . . . .	137
Statistics of Fit . . . . .	137
Data Set Input/Output . . . . .	137
ODS Table Names . . . . .	145
ODS Graphics . . . . .	146
Examples: HPFENGINE Procedure . . . . .	<b>148</b>
Example 4.1: The TASK Option . . . . .	148
Example 4.2: Different Types of Input . . . . .	151
Example 4.3: Incorporating Events . . . . .	154
Example 4.4: Using the SCORE Statement . . . . .	158

Example 4.5: HPFENGINE and HPFDIAGNOSE Procedures . . . . .	163
Example 4.6: The ADJUST Statement . . . . .	166
Example 4.7: Multiple Repositories . . . . .	168
Example 4.8: ODS Graphics . . . . .	169
References . . . . .	<b>173</b>

---



---

## Overview: HPFENGINE Procedure

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 or a default 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, and 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: HPFENGINE Procedure

In its 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.HPFDFTL.

The results of the automatic model selection are displayed in [Output 4.1](#). A summary of the forecast is shown in [Output 4.2](#).

**Figure 4.1** Selection 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

**Figure 4.2** Forecast Summary

Forecast Summary							
Variable	Value	JAN1961	FEB1961	MAR1961	APR1961	MAY1961	JUN1961
AIR	Predicted	445.2972	418.1426	464.0889	494.0261	504.9584	572.5947
Forecast Summary							
Variable	JUL1961	AUG1961	SEP1961	OCT1961	NOV1961	DEC1961	
AIR	662.7040	653.7742	545.8935	487.7147	415.2594	459.6067	

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 as follows to specify the models.

```
proc hpfarimaspec repository=sasuser.repository
    name=arima1;
    dependent symbol=air q=1
        diflist=(1 12) noint transform=log;
run;

proc hpfarimaspec repository=sasuser.repository
    name=arima2;
    dependent symbol=air q=(1,12)
        diflist=(1 12) noint transform=log;
run;
```

The model specifications are grouped together in a selection list. Selection lists are built with the HPFSELECT procedure.

```
proc hpfselect repository=sasuser.repository
    name=myselect;
    spec arima1 arima2;
run;
```

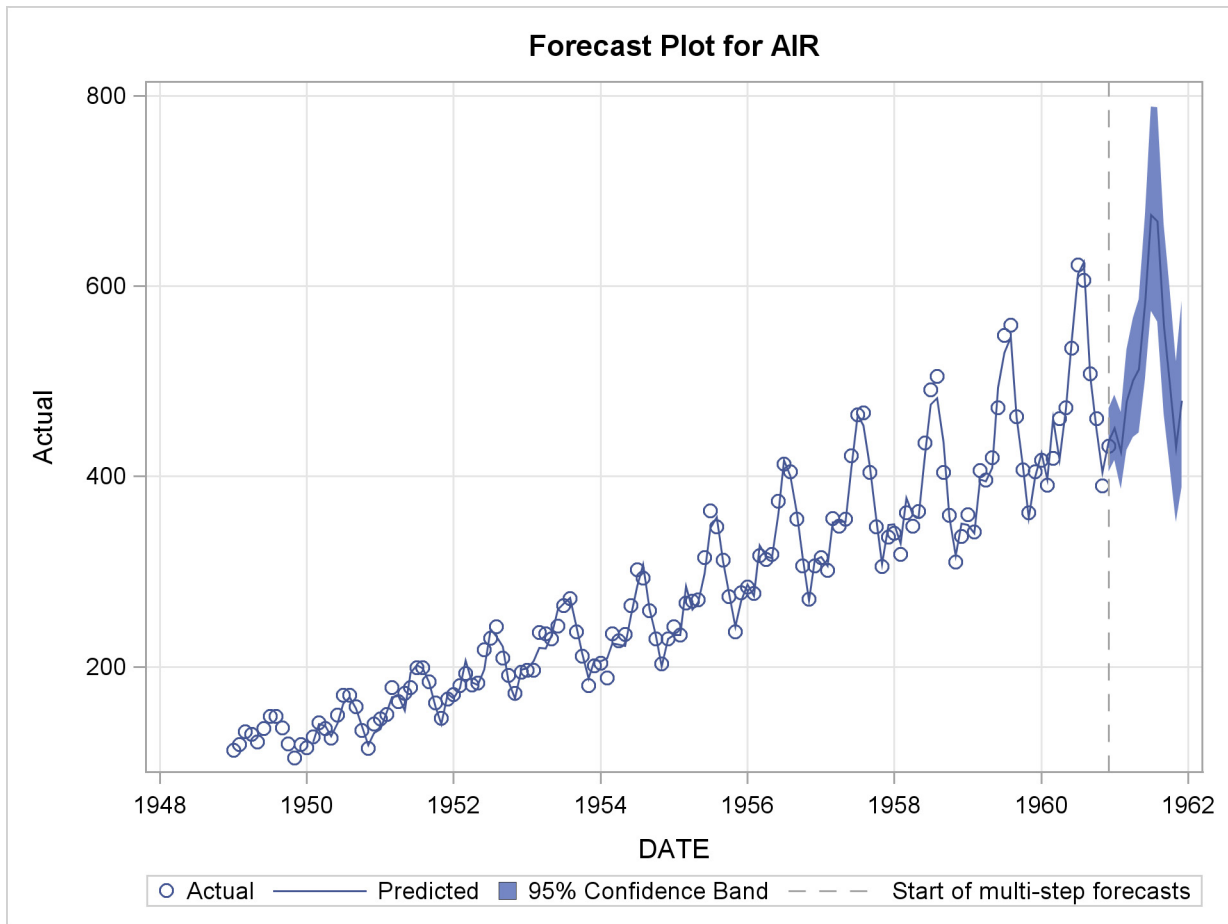
The HPFENGINE procedure uses the GLOBALSELECTION= option to reference the selection list containing the two ARIMA models. Selection results are shown in [Output 4.3](#), and the forecast plot is shown in [Figure 4.4](#).

```
proc hpfengine data=sashelp.air
    repository=sasuser.repository
    globalselection=myselect
    print=(select forecasts)
    plot=forecasts;
    id date interval=month;
    forecast air;
run;
```

**Figure 4.3** Selection and Forecast Results

The HPFENGINE Procedure			
Model Selection Criterion = MAPE			
Model	Statistic	Selected	Label
ARIMA1	3.2553815	No	ARIMA: Log( AIR ) ~ D = (1,12) Q = 1 NOINT
ARIMA2	2.9672282	Yes	ARIMA: Log( AIR ) ~ D = (1,12) Q = (1,12) NOINT

Figure 4.4 Forecasts




---

## Syntax: HPFENGINE Procedure

The following statements are used with the HPFENGINE procedure:

```

PROC HPFENGINE options ;
  ADJUST variable = ( variable-list ) / options ;
  BY variables ;
  CONTROL variable-list / options ;
  EXTERNAL variable-list / options ;
  FORECAST variable-list / options ;
  ID variable INTERVAL= interval options ;
  INPUT variable-list / options ;
  SCORE ;
  STOCHASTIC variable-list / options ;

```

## Functional Summary

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

Statement	Description	Option
<b>Statements</b>		
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	
<b>Model Selection Options</b>		
specify location of model repository	PROC HPFENGINE	REPOSITORY=
specify model selection list	PROC HPFENGINE	GLOBALSELECTION=
<b>Data Set Options</b>		
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	INEVENT=
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	OUTFOR=
specify the forecast component output data set	PROC HPFENGINE	OUTCOMPONENT=
specify the forecast model statistics of fit output data set	PROC HPFENGINE	OUTSTAT=
specify the candidate model statistics of fit output data set	PROC HPFENGINE	OUTSTATSELECT=
specify the input variable output data set	PROC HPFENGINE	OUTINDEP=
specify the detailed model information output data set	PROC HPFENGINE	OUTMODELINFO=
specify the forecast procedure run information output data set	PROC HPFENGINE	OUTPROCINFO=
replace missing values	PROC HPFENGINE	REPLACEMISSING
do not replace missing values in OUTFOR=	PROC HPFENGINE	NOREPLACE
<b>Accumulation Options</b>		
specify the accumulation frequency	ID	INTERVAL=

**Table 4.1** *continued*

<b>Statement</b>	<b>Description</b>	<b>Option</b>
specify the length of seasonal cycle	PROC HPFENGINE	SEASONALITY=
specify the interval alignment	ID	ALIGN=
specify the starting time ID value	ID	START=
specify the ending time ID value	ID	HORIZONSTART=
specify the starting time ID of forecast horizon	ID	END=
specify the date format	ID	FORMAT=
specify the accumulation statistic	ID, FORECAST, INPUT, STOCHASTIC, CONTROL, EXTERNAL	ACCUMULATE=
specify the missing value interpretation	ID, FORECAST, INPUT, STOCHASTIC, CONTROL, EXTERNAL	SETMISSING=
specify the zero value interpretation	ID, FORECAST, INPUT, STOCHASTIC, CONTROL, EXTERNAL	ZEROMISS=
specify the trim missing values	ID, FORECAST, INPUT, STOCHASTIC, CONTROL, EXTERNAL	TRIMMISS=
<b>Forecasting Horizon Options</b>		
specify data to hold back	PROC HPFENGINE	BACK=
specify forecast horizon or lead	PROC HPFENGINE	LEAD=
<b>Forecasting Control Options</b>		
specify forecasting control options	PROC HPFENGINE	TASK=
<b>Scoring Options</b>		
specify location of score repository	PROC HPFENGINE	SCOREREPOSITORY=
<b>Printing and Plotting Options</b>		
specify graphical output	PROC HPFENGINE	PLOT=
specify printed output	PROC HPFENGINE	PRINT=

Table 4.1 *continued*

Statement	Description	Option
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.HPFDFTL.

#### **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 or TASK=UPDATE options are 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, a 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 PROC HPFENGINE 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 useful for evaluating how well the model fits the series. The statistics of fit are based on the entire range of the time series.

**OUTSTATSELECT=** *SAS-data-set*

names the output data set to contain statistics of fit for all of the candidate models fit during model selection. The OUTSTATSELECT= data set is useful for comparing the performance of various models.

**OUTINDEP=** *SAS-data-set*

names the output data set to contain input in the forecasting process. This information is useful if future values of input variables are automatically supplied by the HPFENGINE procedure. Such a case would occur if one or more input variables are listed in either the STOCHASTIC or CONTROLLABLE statement and if there are missing future values of these input variables.

**OUTMODELINFO=** *SAS-data-set*

names the output data set to contain detailed information about the selected forecast model. The data set has information such as the model family, presence or absence of inputs, events and outliers, and so forth.

**OUTPROCINFO=** *SAS-data-set*

names the output data set to contain information in the SAS log, specifically the number of notes, errors, and warnings and the number of series processed, forecasts requested, and forecasts failed.

**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.

**PRINT=** *option | (options)*

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.HPFDFTL.

### 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 be 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
MAPES	Mean Absolute Error Percent of Standard Deviation
MDAPES	Median Absolute Error Percent of Standard Deviation
GMAPES	Geometric Mean Absolute Error Percent of Standard Deviation
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
MASE	Mean Absolute Scaled Error
RSQUARE	R-Square
ADJRSQ	Adjusted R-Square
AADJRSQ	Amemiya's Adjusted R-Square
RWRSQ	Random Walk R-Square
AIC	Akaike Information Criterion
AICC	Finite Sample Corrected AIC
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 SEASONTTEST=(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.HPFDFTL.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* )**

enables 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 be printed.

Available *stage-options* are as follows:

PROCEDURELEVEL specifies that the procedure stage be 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)
```

---

## 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 ADJUST statement, missing values are set based on the ZEROMISS= option of the ID statement. See the ID statement ZEROMISS= option for more details.

---

## 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.

---

## 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.

---

## 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.

---

## 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.

### **NOREPLACE**

specifies that the forecast value in the OUTFOR= data set be set to missing when a corresponding historical value is missing. For example, consider historical data spanning 01JAN1980 through 01DEC1985. If the dependent variable corresponding to 01MAR1983 is missing, the forecast variable in the OUTFOR= data set at time ID 01MAR1983 is usually replaced by the one-step-ahead forecast. The NOREPLACE option prevents that one-step-ahead forecast from being written in OUTFOR=. The inverse behavior of this option, applied to the OUT= data set, is REPLACEMISSING.

### **REPLACEMISSING**

specifies that embedded missing actual values be replaced with one-step-ahead forecasts in the OUT= data set. The inverse behavior of this option, applied to the OUTFOR= data set, is NOREPLACE.

### **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**. The default is **BEGINNING**.

**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.

**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.

**HORIZONSTART=** *option*

specifies a SAS date, datetime, or time value that represents the start of the forecast horizon. If the specified **HORIZONSTART=** date falls beyond the end of the historical data, then forecasts are computed from the last observation with a nonmissing dependent variable value until **LEAD=** intervals from the **HORIZONSTART=** data. Therefore, the effective forecast horizon for any particular **BY** group may differ from another due to differences in the lengths of the historical data across the **BY** groups, but all forecasts will end at the same date as determined by the **HORIZONSTART=** and **LEAD=** options.

An important feature of the **HORIZONSTART=** option is that it will only truncate values in forecast variables. Any future values of input variables will be retained.

**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 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. 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.

---

## 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=.

---

## 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.

---

## Details: HPFENGINE Procedure

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:

```
01MAR1999    40
01APR1999    .
01MAY1999    90
```

If the `ACCUMULATE=AVERAGE` option is specified:

```
01MAR1999    20
01APR1999    .
01MAY1999    30
```

If the `ACCUMULATE=MINIMUM` option is specified:

```
01MAR1999    10
01APR1999    .
01MAY1999    20
```

If the `ACCUMULATE=MEDIAN` option is specified:

```

O1MAR1999    20
O1APR1999    .
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  $i = 1, \dots, M$  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  $i = 1, \dots, M$ .

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  $i = M, \dots, 1$ , 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,  $op_i^a = \text{inverse}(op_i^b)$ .

For example, if the pre-adjustment operator is subtraction, the post-adjustment operator is addition, as shown in the following:

$$w_t = y_t - \sum_{i=1}^M x_{i,t}$$

$$\hat{y}_t = \hat{w}_t + \sum_{i=1}^M x_{i,t}$$

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 SEASONTTEST= option in the HPFSELECT procedure's DIAGNOSE statement sets the threshold for categorizing a series as seasonal or nonseasonal.

These diagnostic categorizations are used during the model selection process to ensure only appropriate models are fit to a given series.

---

## 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= option is specified, or the one-step-ahead forecasts for the full range of the time series if the HOLDOUT= and HOLDOUTPCT= option is not specified.

The candidate model with the best selection criterion is selected to forecast the time series.

Not all models specifications in a model selection list are necessarily used as candidates during the selection process. Diagnostic tests may tag some model specifications as unsuitable. Generally, if a time series is judged as seasonal by the diagnostic tests, only seasonal models in the model selection list are candidates. Likewise, if a time series is nonseasonal, only nonseasonal models are fit. If the seasonal characteristics of a time series do not match those of any models in a selection list, the seasonal diagnostic test is turned off and the selection process restarted.

Characteristics that make a model seasonal vary according to the family of model. Those characteristics are listed here by model family.

Smoothing models	The Winters, additive Winters, and seasonal smoothing models are considered seasonal. Linear, simple, double, and damped trend models smoothing are considered nonseasonal.
ARIMA models	An ARIMA model is considered seasonal if there is a difference of order equal to the seasonal cycle length of the time ID. The presence of MA or AR terms with lags equal to the seasonal cycle length also qualifies a model as seasonal. The presence of a predefined seasonal input is another factor that tags a model as seasonal.
UCM	An unobserved component model is seasonal if there is a seasonal component present, specified by the SEASON statement in the HPFUCM-SPEC procedure, or if there is a predefined seasonal input.

Similar subsetting of selection lists occurs after an intermittency diagnostic test. The HPFENGINE procedure avoids direct comparison of results from intermittent models and nonintermittent models. If the diagnostic tests deem a time series intermittent, only intermittent models will be used as candidates during the selection process. If a series is nonintermittent, only nonintermittent series will be used.

---

## 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. The starting point for parameter estimation, either automatically chosen by the optimizer or supplied programmatically, can be controlled using the TASK= option.

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. The ZEROMISS= option is used to replace zero values with missing values at either the beginning or end of a series. In such cases the SETMISSING= option is applied after the ZEROMISS= option.

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. Options such as HORIZONSTART= may implicitly add missing values to the end of a series and also cause forecast horizons greater than that specified by LEAD=.

---

## 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 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 following input data sets supply series data, selection list mapping information and/or parameter estimates, and event data base information, respectively:

- DATA=
- INEST=
- INEVENT=

Additionally, the HPFENGINE procedure can create the following data sets:

- OUT=
- OUTFOR=

- OUTEST=
- OUTCOMPONENT=
- OUTSTAT=
- OUTSTATSELECT=
- OUTMODELINFO=
- OUTINDEP=
- OUTPROCINFO=

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 supplies mapping information associating individual time series with model selection lists. It may optionally include information about model parameter estimates. Parameter estimates may be required depending on the setting of the TASK= option. This data set is typically created by either the HPFDIAGNOSE procedure or a prior invocation of the HPFENGINE procedure with the OUTEST= option.

The INEST= data set is not required. If not present, a model selection list can be supplied for use by all dependent series using the GLOBALSELECTION= option. In the absence of both the INEST= data set and the GLOBALSELECTION= option, a default selection list is used.

If the INEST= data set is supplied but there are some forecast variables that have no mapping in INEST=, those unmatched variables are forecast using model specifications found either using the GLOBALSELECTION= option or in the default list.

The INEST= data set contains the variables specified in the BY statement as well as the following variables:

<code>_NAME_</code>	Variable name
<code>_SELECT_</code>	Name of selection list
<code>_MODEL_</code>	Name of model
<code>_MODELVAR_</code>	Model variable mapping
<code>_DSVAR_</code>	Data set variable mapping

The referenced selection lists, taken together with the data set to model variable mappings, drive the forecasting process.

If the HPFENGINE statement TASK= option is specified and set to either FORECAST or UPDATE, other variables should be present in INEST=. The additional variables are as follows:

<code>_TRANSFORM_</code>	Transformation applied
<code>_COMPONENT_</code>	Model component (e.g., AR, MA, trend, etc.)
<code>_COMPMODEL_</code>	Model portion of an intermittent demand component
<code>_FACTOR_</code>	Model factor
<code>_LAG_</code>	Lag of input
<code>_SHIFT_</code>	Shift
<code>_PARAM_</code>	Parameter name
<code>_LABEL_</code>	Parameter label
<code>_EST_</code>	Parameter estimate
<code>_STDERR_</code>	Parameter estimate standard error
<code>_TVALUE_</code>	Parameter estimate <i>t</i> -value
<code>_PVALUE_</code>	Parameter estimate <i>p</i> -value
<code>_STATUS_</code>	Indicates success/failure in estimating parameter

### INEVENT= Data Set

The INEVENT= contains information describing predefined events. This data set is usually created by the HPFEVENTS procedure and is required only if events are included in a model. Details are found in the documentation of the HPFEVENTS procedure.

### 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:

<code>_NAME_</code>	Variable name
<code>_TIMEID_</code>	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.

Procedures that define model specifications have TRANSFORM= and MEDIAN options used to apply functional transformations to series and the subsequent inverse transformation of forecast results. 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.)
_COMPMODEL_	Model portion of an intermittent demand component
_FACTOR_	Model factor
_LAG_	Lag of input
_SHIFT_	Shift
_PARAM_	Parameter name
_LABEL_	Parameter label
_EST_	Parameter estimate
_STDERR_	Parameter estimate standard error
_TVALUE_	Parameter estimate <i>t</i> -value
_PVALUE_	Parameter estimate <i>p</i> -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. If the option TASK=UPDATE is used, the parameters are estimated again, this time using the supplied estimates as starting values for the optimization process.

## OUTCOMPONENT= Data Set

The contents of the OUTCOMPONENT set vary depending upon the forecast model. See Chapter 14, “[Forecasting Process Details](#),” for information about specific model components.

The OUTCOMPONENT= data set contains the variables specified in the BY statement as well as the following variables:

<code>_NAME_</code>	Variable name
<code>_COMP_</code>	Name of the component
<code>_TIME_</code>	Time ID
<code>_ACTUAL_</code>	Dependent series value
<code>_PREDICT_</code>	Component forecast
<code>_LOWER_</code>	Lower confidence limit
<code>_UPPER_</code>	Upper confidence limit
<code>_STD_</code>	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:

<code>_NAME_</code>	Variable name
<code>_REGION_</code>	Indicates the region in which the statistics are calculated. Statistics calculated in the fit region are indicated by FIT. Statistics calculated in the forecast region, which happens only if the BACK= option is greater than zero, are indicated by FORECAST.
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
TSS	Total Sum of Squares
SST	Corrected Total Sum of Squares
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
MASE	Mean Absolute Scaled Error
RSQUARE	R-Square
ADJRSQ	Adjusted R-square
AADJRSQ	Amemiya's Adjusted R-Square
RWRSQ	Random Walk R-Square
AIC	Akaike Information Criterion
AICC	Finite Sample Corrected AIC
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
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
MAPES	Mean Absolute Error Percent of Standard Deviation
MDAPES	Median Absolute Error Percent of Standard Deviation
GMAPES	Geometric Mean Absolute Error Percent of Standard Deviation

If the statistics-of-fit step fails for a particular variable, no observations are recorded.

### OUTSTATSELECT= Data Set

The OUTSTATSELECT= data set contains the same variables as the OUTSTAT= data set with the addition of the following:

<code>_MODEL_</code>	Model specification name
<code>_SELECT_</code>	Name of model selection list to which <code>_MODEL_</code> belongs
<code>_SELECTED_</code>	Indicates whether or not this model was chosen to forecast the dependent series

### OUTINDEP= Data Set

The OUTINDEP= data set contains data used as input in the forecasting process. This information is useful if future values of input variables are automatically supplied by the HPFENGINE procedure. Such a case would occur if one or more input variables are listed in either the STOCHASTIC or CONTROLLABLE statement and if there are missing future values of these input variables.

The OUTINDEP= data set contains the variables specified in the BY statement as well as the following variables:

<code>_NAME_</code>	Variable name
<code>_TIMEID_</code>	Time ID values
<code>_X_</code>	Values of the input variable <code>_NAME_</code>

### OUTMODELINFO= Data Set

The OUTMODELINFO= data set provides information about the selected forecast model and contains the following variables.

<code>_NAME_</code>	Variable name
<code>_MODEL_</code>	Model specification name
<code>_MODELTYPE_</code>	Model specification type, either ARIMA, ESM, UCM, IDM, EXTERNAL, or INACTIVE

<code>_DEPTRANS_</code>	Name of transform applied to dependent variable or NONE if no transform
<code>_SEASONAL_</code>	Set to 1 if the model is seasonal, 0 otherwise
<code>_TREND_</code>	Set to 1 if the model has a trend, 0 otherwise
<code>_INPUTS_</code>	Set to 1 if one or more inputs are present, 0 otherwise
<code>_EVENTS_</code>	Set to 1 if one or more events are present, 0 otherwise
<code>_OUTLIERS_</code>	Set to 1 if one or more outliers are present, 0 otherwise

Characteristics that make a model seasonal vary according to the family of model. Those characteristics are listed here by model family.

Smoothing models	The Winters, additive Winters, and seasonal smoothing models are considered seasonal. Linear, simple, double, and damped trend models smoothing are considered nonseasonal.
ARIMA models	An ARIMA model is considered seasonal if there is a difference with order equal to the seasonal cycle length of the time ID. The presence of MA or AR terms with lags equal to the seasonal cycle length also qualifies a model as seasonal. The presence of a predefined seasonal input is another factor that tags a model as seasonal.
UCM	An unobserved component model is seasonal if there is a seasonal component present, specified by the SEASON statement in the HPFUCM-SPEC procedure, or if there is a predefined seasonal input.
IDM	Intermittent demand models are always considered nonseasonal.

Likewise, characteristics of a model that indicate a trend vary according to the family of model.

Smoothing models	The simple and seasonal smoothing models do not have a trend. Linear, double, damped trend models, and multiplicative and additive Winters models do have a trend.
ARIMA models	An ARIMA model is considered to have a trend if there is a first- or second- order difference applied to the dependent variable. A predefined trend in an input statement also qualifies the model as having a trend component.
UCM	An unobserved component model has a trend if there is a slope component or a predefined trend in an input statement.
IDM	Intermittent demand models do have a trend.

## OUTPROCINFO= Data Set

The OUTPROCINFO= data set contains information about the run of the HPFENGINE procedure. The following variables are present:

<code>_SOURCE_</code>	Set to the name of the procedure, in this case HPFENGINE
-----------------------	--

<code>_NAME_</code>	Name of an item being reported; may be the number of errors, notes, or warnings, number of forecasts requested, etc.
<code>_LABEL_</code>	Descriptive label for the item in <code>_NAME_</code>
<code>_STAGE_</code>	Set to the current stage of the procedure, for HPFENGINE this is set to ALL
<code>_VALUE_</code>	Value of the item specified in <code>_NAME_</code>

## ODS Table Names

Table 4.2 relates the PRINT= options to ODS tables.

**Table 4.2** ODS Tables Produced in PROC HPFENGINE

ODS Table Name	Description	Specific Models
<b>ODS Tables Created by the PRINT=SUMMARY Option</b>		
Variable	Forecast Variable Information	
ForecastSummary	Forecast Summary	
<b>ODS Tables Created by the PRINT=ESTIMATES Option</b>		
Variable	Forecast Variable Information	
ParameterEstimates	Parameter Estimates	
<b>ODS Tables Created by the PRINT=SELECT Option</b>		
Variable	Forecast Variable Information	
ModelSelection	Model Selection Statistics	
<b>ODS Tables Created by the PRINT=FORECASTS Option</b>		
Variable	Forecast Variable Information	
Forecasts	Forecast	
Demands	Demands	IDM models only
DemandSummary	Demand Summary	IDM models only
<b>ODS Tables Created by the PRINT=STATISTICS Option</b>		
Variable	Forecast Variable Information	
FitStatistics	Statistics of Fit	
<b>ODS Tables Created by the PRINT=BIAS Option</b>		
Variable	Forecast Variable Information	

**Table 4.2** *continued*

ODS Table Name	Description	Specific Models
TestUnbiasedness	Bias Test	
ParameterEstimates	Bias Test Parameter Estimates	
<b>ODS Tables Created by the PRINT=CANDIDATES Option</b>		
Variable	Forecast Variable Information	
ParameterEstimates	Parameter Estimates	
<b>ODS Tables Created by the PRINT=COMPONENTS Option</b>		
Variable	Forecast Variable Information	
ComponentEstimates	Parameter Estimates	
<b>ODS Tables Created by the PRINT=PERFORMANCE Option</b>		
Variable	Forecast Variable Information	
Performance	Performance Statistics	
<b>ODS Tables Created by the PRINT=PERFORMANCESUMMARY Option</b>		
Variable	Forecast Variable Information	
PerformanceSummary	Performance Summary	
<b>ODS Tables Created by the PRINT=PERFORMANCEOVERALL Option</b>		
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

This section describes the use of ODS for creating graphics with the HPFENGINE procedure.

To request these graphs, you must specify the ODS GRAPHICS statement. In addition, you can specify the **PLOT=** option in the PROC HPFENGINE statement.

## 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 4.3](#).

To request these graphs, you must specify the ODS GRAPHICS statement. In addition, you must specify the [PLOT=](#) option in the PROC HPFENGINE statement.

**Table 4.3** ODS Graphs Produced by PROC HPFENGINE

ODS Graph Name	Plot Description	Statement	PLOT= Option
CandidateErrorHoldoutPlot	Plot of Candidate Model Errors with Holdout	PROC HPFENGINE	CANDIDATES
CandidateErrorPlot	Plot of Candidate Model Errors	PROC HPFENGINE	CANDIDATES
CandidateModelHoldoutPlot	Plot of Candidate Models with Holdout	PROC HPFENGINE	CANDIDATES
CandidateModelPlot	Plot of Candidate Models	PROC HPFENGINE	CANDIDATES
ComponentEstimatesPlot	Plot of Component Estimates	PROC HPFENGINE	COMPONENTS
DemandErrorsPlot	Average Demand Errors	PROC HPFENGINE	ERRORS
DemandForecastsPlot	Average Demand Forecasts	PROC HPFENGINE	FORECASTS
DemandIntervalHistogram	Demand Interval Histogram	PROC HPFENGINE	ALL
DemandIntervalPlot	Demand Interval Forecast Plot	PROC HPFENGINE	ALL
DemandSizeHistogram	Demand Size Histogram	PROC HPFENGINE	MODELS
DemandSizePlot	Demand Size Forecast Plot	PROC HPFENGINE	MODELS
ErrorACFNORMPlot	Standardized autocorrelation of Prediction Errors	PROC HPFENGINE	ACF
ErrorACFPlot	Autocorrelation of Prediction Errors	PROC HPFENGINE	ACF
ErrorHistogram	Prediction Error Histogram	PROC HPFENGINE	ERRORS
ErrorIACFNORMPlot	Standardized Inverse Autocorrelation of Prediction Errors	PROC HPFENGINE	IACF
ErrorIACFPlot	Inverse Autocorrelation of Prediction Errors	PROC HPFENGINE	IACF

Table 4.3 continued

ODS Graph Name	Plot Description	Statement	Option
ErrorPACFNORMPlot	Standardized Partial Autocorrelation of Prediction Errors	PROC HPFENGINE	PACF
ErrorPACFPlot	Partial Autocorrelation of Prediction Errors	PROC HPFENGINE	PACF
ErrorPlot	Plot of Prediction Errors	PROC HPFENGINE	ERRORS
ErrorWhiteNoiseLogProbPlot	White Noise Log Probability Plot of Prediction Errors	PROC HPFENGINE	WN
ErrorWhiteNoisePlot	White Noise Plot of Prediction Errors	PROC HPFENGINE	ALL
ErrorWhiteNoiseProbPlot	White Noise Probability Plot of Prediction Errors	PROC HPFENGINE	WN
ForecastsOnlyPlot	Forecasts Only Plot	PROC HPFENGINE	FORECASTSONLY
ForecastsPlot	Forecasts Plot	PROC HPFENGINE	FORECAST
ModelForecastsPlot	Model and Forecasts Plot	PROC HPFENGINE	ALL
ModelPlot	Model Plot	PROC HPFENGINE	ALL
StockingAveragePlot	Stocking Average Plot	PROC HPFENGINE	FORECASTS
StockingLevelPlot	Stocking Level Plot	PROC HPFENGINE	FORECASTS

---

## Examples: HPFENGINE Procedure

---

### Example 4.1: The TASK Option

The default selection list is used in this example. The first call to the HPFENGINE procedure which follows uses the default TASK = SELECT action. A model is selected from the default list, parameters are estimated, and a forecast is produced. Selection results are shown in [Output 4.1.1](#).

```
proc hpfengine data=sashelp.citimon
              outfor=outfor
              print=select;
  id date interval=month;
  forecast eec;
run;
```

**Output 4.1.1** 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	0.16293237	No	Winters Method (Additive)
smwint	0.17985708	No	Winters Method (Multiplicative)
smseas	0.16291415	Yes	Seasonal Exponential Smoothing

The second call to the HPFENGINE procedure follows. It demonstrates how to use the same model specifications but alter the selection criterion and add a holdout. It is not necessary to create a new selection list to make these changes, since the TASK = SELECT option can override the settings in an existing list. Selection results are shown in [Output 4.1.2](#).

```
proc hpfengine data=sashelp.citimon
    outfor=outfor
    outest=outest
    print=select
    task=select(criterion=mape holdout=24 override);
    id date interval=month;
    forecast eec;
run;
```

**Output 4.1.2** Selection and Forecast Results

The HPFENGINE Procedure			
Model Selection Criterion = MAPE			
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	2.2881620	Yes	Winters Method (Additive)
smwint	2.9441207	No	Winters Method (Multiplicative)
smseas	2.5775447	No	Seasonal Exponential Smoothing

Perhaps there are revisions to recent historical data and there is a need to forecast using the updated information but with the same model used to forecast earlier data. This is done easily with the TASK = UPDATE option. The following DATA step is used to simulate revision to the data.

```
data citimon;
```

```

set sashelp.citimon;
if date ge '01JAN1991'd then eec = eec + 0.1;
run;

```

In the following statements, the HPFENGINE procedure is called with TASK=UPDATE and the additional instruction to change the confidence interval to 90model is the same used in the previous call of the HPFENGINE procedure, and a reference to this model, together with estimates of its parameters, is found in the OUTEST= data set. The parameters are estimated again using the full range of data, and the prior parameter estimates are used as starting points in the optimization of the new estimates.

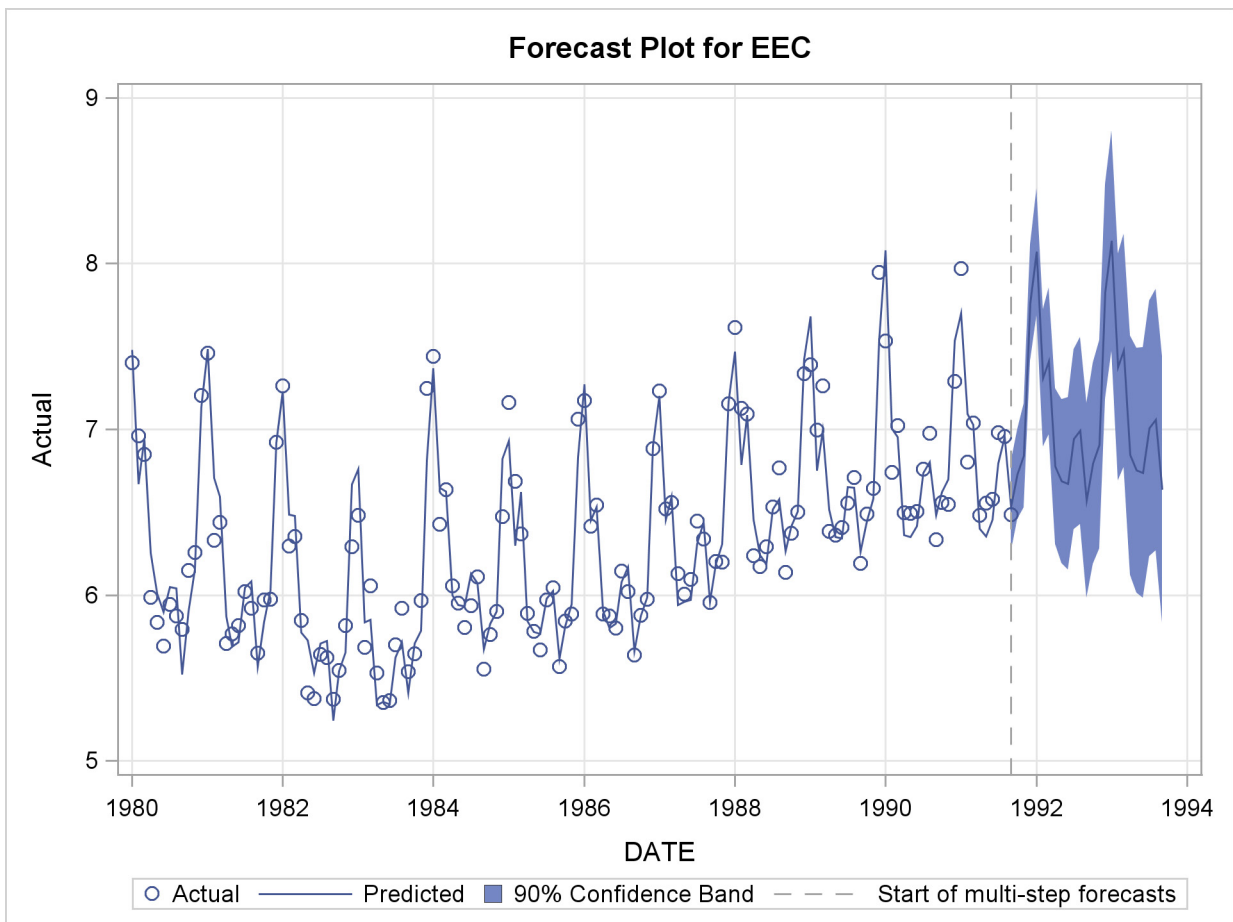
The forecast plot is shown in [Figure 4.1.3](#).

```

proc hpfengine data=citimon
  outfor=outfor
  inest=outest
  plot=forecasts
  lead=24
  task=update(alpha=.1 override);
  id date interval=month;
  forecast eec;
run;

```

#### Output 4.1.3 Forecasts





## Example 4.2: Different Types of Input

This example demonstrates the use of different input types in the HPFENGINE procedure.

The following statements read Input Gas Rate and Output CO2 from a gas furnace. (Data values are not shown. See “Series J” in (Box, Jenkins, and Reinsel 1994) for the values.)

```
data seriesj;
  input x y @@;
  label x = 'Input Gas Rate'
        y = 'Output CO2';
  date = intnx( 'day', '01jan1950'd, _n_-1 );
  format Date DATE.;
datalines;

... more lines ...
```

Begin by creating an ARIMA model specification using the HPFARIMASPEC procedure as follows. The new model specification is then placed into a model selection list using the HPFSELECT procedure.

```
proc hpfarimaspec repository=sasuser.repository
                  name=arimaspc;
  dependent symbol=Y p=2;
  input      symbol=X num=2 den=1 lag=3;
run;

proc hpfselect repository=sasuser.repository
              name=myselect;
  spec arimaspc;
run;
```

There are no future values of the independent variables given in the seriesj data set. For this example a DATA step is used as follows to set the last few observations of the dependent value to missing. Values of the independent variable are left intact. This ensures that some future values of the independent variable are available for forecasting the dependent variable between 17OCT1950 and the end of the data set.

```
data seriesj_trunc;
  set seriesj;
  if (date >= '17oct1950'd) then y = .;
run;
```

In the following statements, the HPFENGINE procedure is called using the INPUT statement to identify the data set variable “x” as input. No missing future values, even if required, will be computed for “x”. The OUTINDEP= data set contains values of the input variable used in the forecast.

```
proc hpfengine data=seriesj_trunc
              outfor=outfor
```

```

        outindep=outindep
        repository=sasuser.repository
        globalselection=myselect
        lead=7;
    id date interval=day;
    forecast y;
    input    x;
run;

proc print data=outindep(where=(date >= '17oct1950'd)) label noobs;
var date x;
run;

proc print data=outfor(where=(date >= '17oct1950'd)) label noobs;
    var date predict upper lower;
run;

```

The user-supplied future values of the input variable used in the forecast as well as the forecasts themselves are shown in [Output 4.2.1](#) and [Output 4.2.2](#).

#### Output 4.2.1 Future Values of X Supplied by the User

date	Input Values
17OCT1950	0.204
18OCT1950	0.253
19OCT1950	0.195
20OCT1950	0.131
21OCT1950	0.017
22OCT1950	-0.182
23OCT1950	-0.262

#### Output 4.2.2 Forecasts for Y

date	Predicted Values	Upper Confidence Limits	Lower Confidence Limits
17OCT1950	57.0684	57.5116	56.6252
18OCT1950	56.4050	57.1794	55.6307
19OCT1950	55.3432	56.3382	54.3481
20OCT1950	54.1846	55.2925	53.0767
21OCT1950	53.2290	54.3758	52.0821
22OCT1950	52.6229	53.7751	51.4706
23OCT1950	52.3790	53.5317	51.2263

To demonstrate use of the STOCHASTIC statement, a DATA step is used as follows to eliminate future values of the input variable.

```

data seriesj_trunc;
    set seriesj;

```

```

    if (date < '17oct1950'd);
run;

```

In the following statements, the HPFENGINE procedure identifies the input variable using the STOCHASTIC statement and automatically forecasts the input variable.

```

proc hpfengine data=seriesj_trunc
    outfor=outfor
    outindep=outindep
    repository=sasuser.repository
    globalselection=myselect
    lead=7;
    id date interval=day;
    forecast y;
    stochastic x;
run;

proc print data=outindep(where=(date >= '17oct1950'd)) label noobs;
var date x;
run;

proc print data=outfor(where=(date >= '17oct1950'd)) label noobs;
    var date predict upper lower;
run;

```

The future values of the input variable, automatically forecast, are shown in [Output 4.2.3](#). The forecasts of the dependent variable are shown in [Output 4.2.4](#).

#### Output 4.2.3 Future Values of X Automatically Forecast

date	Input Values
17OCT1950	0.21222
18OCT1950	0.34557
19OCT1950	0.44535
20OCT1950	0.52002
21OCT1950	0.57588
22OCT1950	0.61769
23OCT1950	0.64897

#### Output 4.2.4 Forecasts for Y

date	Predicted Values	Upper Confidence Limits	Lower Confidence Limits
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

### Example 4.3: 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.

First a DATA step is used to generate a data set with a shift beginning at “01OCT1980”.

```
data shifted;
  set sashelp.workers;
  if date >= '01oct80'd then Y = electric+100;
  else Y = electric;
  drop masonry electric;
run;
```

Next the HPFEVENTS procedure is used to create an events database. The database will contain the definition of an event named “promotion,” a level shift beginning at “01OCT1980.”

```
proc hpfevents data=shifted lead=12;
  id date interval=month;
  eventdef promotion='01oct80'd / TYPE=LS;
  eventdata out= evdsout1;
  eventdummy out= evdumout1;
run;
```

Then two ARIMA model specification are created as follows. Both of them will be added to a selection list for use by the HPFENGINE procedure.

```
proc hpfarimaspec repository=sasuser.repository
  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.repository
  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;
```

The HPFSELECT procedure then creates a new selection list containing the two ARIMA specifications, as follows, and uses the EVENTMAP option to add the “promotion” event to each of them.

```
proc hpfselect repository=sasuser.repository
  name=myselect
  label="My Selection List";
  select select=mape;
  spec sp1 sp2 /
  eventmap(symbol=_NONE_ event=promotion);
run;
```

The HPFENGINE procedure fits both ARIMA models to the data, with the “promotion” event, as follows. The results of the model selection are shown in [Output 4.3.1](#). Parameter estimates for the selected model are shown in [Output 4.3.2](#). The forecast is displayed in [Figure 4.3.3](#).

```
proc hpfengine data=shifted
  globalselection=myselect
  repository=sasuser.repository
  inevent=evdsout1
  print=(select estimates)
  plot=forecasts;
  id date interval=month;
  forecast y;
run;
```

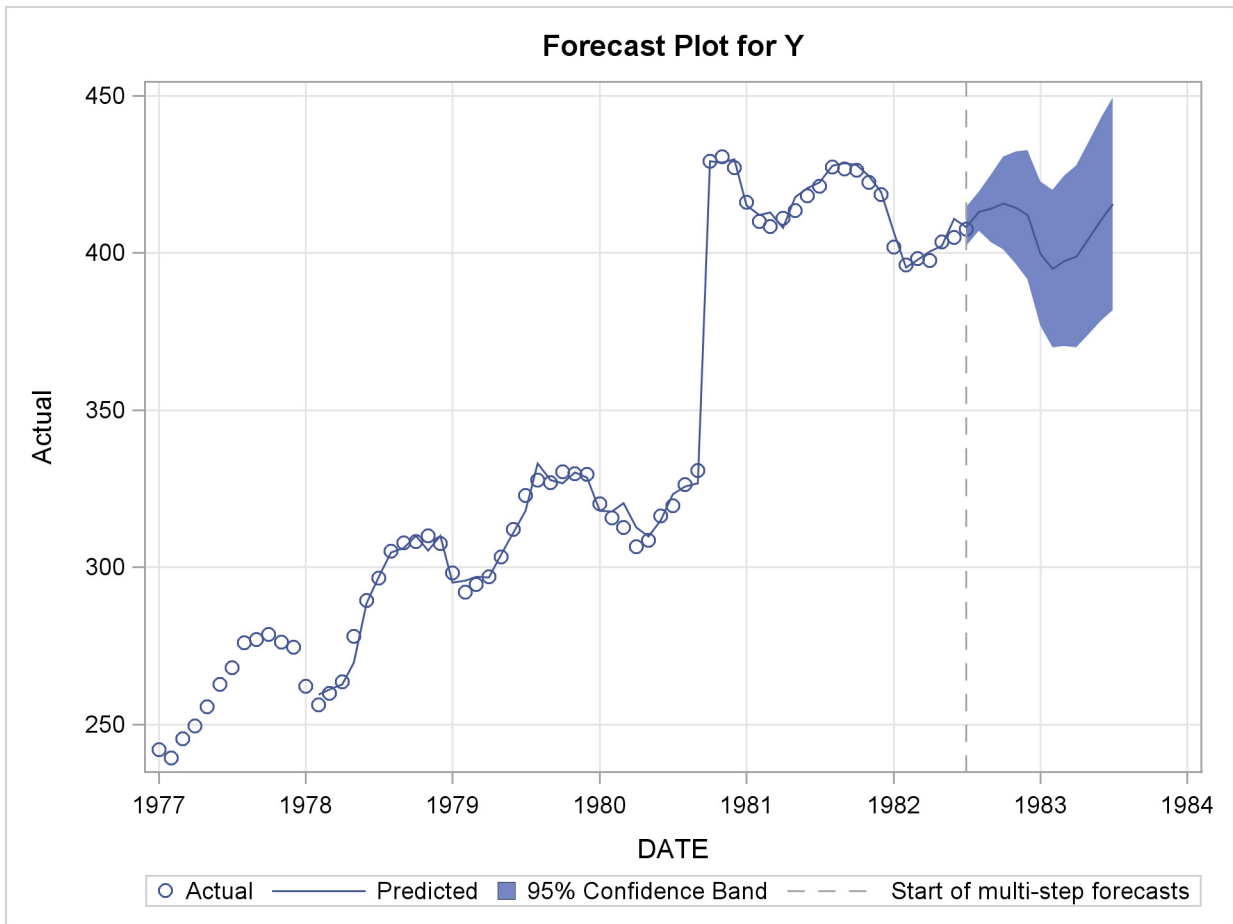
#### Output 4.3.1 Model Selection Results

The HPFENGINE Procedure			
Model Selection Criterion = MAPE			
Model	Statistic	Selected	Label
SP1	0.70600173	Yes	ARIMA(0,1,2) (0,1,1)_12 No Intercept
SP2	0.76609476	No	ARIMA(2,1,0) (1,1,0)_12 No Intercept

#### Output 4.3.2 Parameter Estimates of Selected Model

Parameter Estimates for SP1 Model					
Component	Parameter	Estimate	Standard Error	t Value	Approx Pr >  t
Y	MA1_1	-0.41281	0.14173	-2.91	0.0053
Y	MA1_2	-0.21826	0.14382	-1.52	0.1354
Y	MA2_12	0.80211	0.11064	7.25	<.0001
PROMOTION	SCALE	94.91032	2.91440	32.57	<.0001

## Output 4.3.3 Forecasts



An alternate way to include inputs is to refer to the dummy variable in the data set created by the EVENTDUMMY statement in the HPFEVENTS procedure, as shown in the following code.

The ARIMA models are different in this case because the event data are explicitly added as an input. In order to produce the same results as the EVENTMAP method of handling events, the input variables are differenced in the same manner as the dependent variable.

```
proc hpfarimaspec repository=sasuser.repository
    name=sp1
    label="ARIMA(0,1,2)(0,1,1)_12 No Intercept";
    dependent symbol=Y q=(1,2)(12) diflist=1 12 noint;
    input      symbol=promotion diflist=1 12;
run;

proc hpfarimaspec repository=sasuser.repository
    name=sp2
    label="ARIMA(2,1,0)(1,1,0)_12 No Intercept";
    dependent symbol=Y p=(1,2)(12) diflist=1 12 noint;
    input      symbol=promotion diflist=1 12;
run;
```

Again, the HPFSELECT procedure creates a new selection list containing the two ARIMA specifications, as follows. This time no EVENTMAP option is required.

```
proc hpfselect repository=sasuser.repository
              name=myselect
              label="My Selection List";
  select select=mape;
  spec sp1 sp2;
run;
```

The HPFENGINE procedure will need to find the “promotion” variable in the input data set. A DATA step is used as follows to merge the input variable with the data set containing the dependent variable.

```
data shifted;
  merge shifted evdumout1(drop=y);
  by date;
run;
```

The HPFENGINE procedure fits both ARIMA models to the data, with the “promotion” input, as follows. The results of the model selection are shown in [Output 4.3.4](#). Parameter estimates for the selected model are in [Output 4.3.5](#). The forecast is displayed in [Figure 4.3.6](#). Notice that the results are the same as the results of using the EVENTMAP option in the HPFSELECT procedure. This technique of avoiding the EVENTMAP option and including the event is useful if events need to enter the forecast model through more complex transfer functions.

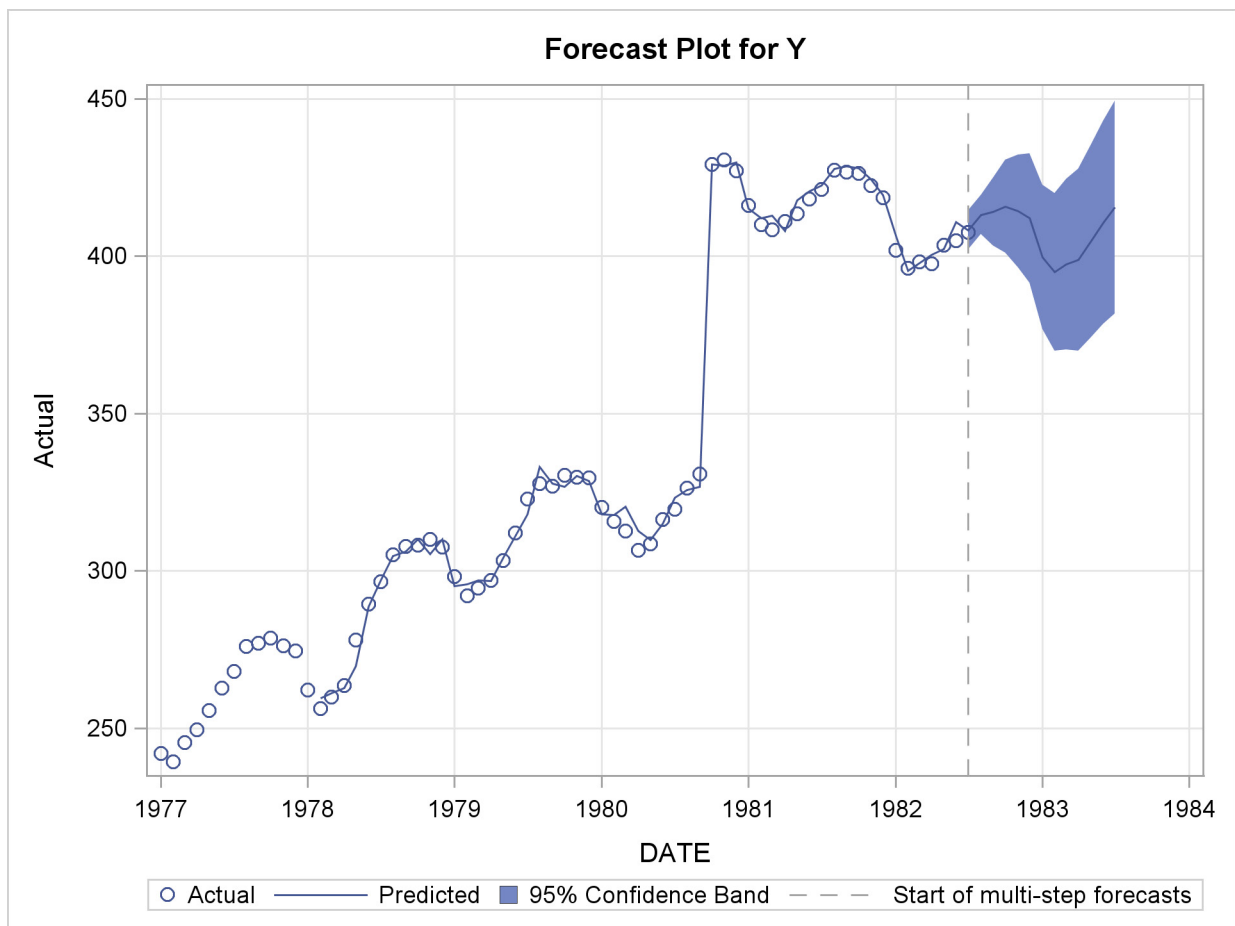
```
proc hpfengine data=shifted
              globalelection=myselect
              repository=sasuser.repository
              inevent=evdsout1
              print=(select estimates)
              plot=forecasts;
  id date interval=month;
  forecast y;
  input promotion;
run;
```

#### Output 4.3.4 Model Selection Results

The HPFENGINE Procedure			
Model Selection Criterion = MAPE			
Model	Statistic	Selected	Label
SP1	0.70600173	Yes	ARIMA(0,1,2) (0,1,1)_12 No Intercept
SP2	0.76609476	No	ARIMA(2,1,0) (1,1,0)_12 No Intercept

**Output 4.3.5** Parameter Estimates of Selected Model

Parameter Estimates for SP1 Model					
Component	Parameter	Estimate	Standard Error	t Value	Approx Pr >  t
Y	MA1_1	-0.41281	0.14173	-2.91	0.0053
Y	MA1_2	-0.21826	0.14382	-1.52	0.1354
Y	MA2_12	0.80211	0.11064	7.25	<.0001
promotion	SCALE	94.91032	2.91440	32.57	<.0001

**Output 4.3.6** Forecasts

### Example 4.4: Using the SCORE Statement

This example demonstrates the use of the SCORE statement to create a forecast score file. Score files are used by the HPFSCSUB function to produce forecasts outside of the HPFENGINE procedure.

In this particular case price and sales data are present. A forecast score file is produced with price as



a controllable input. The NLP procedure is used to maximize a simple expression of total revenue in the forecast horizon, as the optimizer adjusts the price inputs of the forecast function. The input values found by the NLP procedure are then used in the HPFENGINE procedure again to create a forecast plot.

The following DATA step uses data from a single BY group in the SASUSER.PRICEDATA data set.

```
data pricedata;
  set sashelp.pricedata(where=(product=1));
  keep date sale price;
run;
```

An ARIMAX model specification is created as follows, together with a selection list that references this specification.

```
proc hpfarimaspec repository=work.repository name=arimax;
  forecast symbol=sale q=(12) diflist=12 noint;
  input symbol=price diflist=12;
run;

proc hpfselect repository=work.repository name=select;
  spec arimax;
run;
```

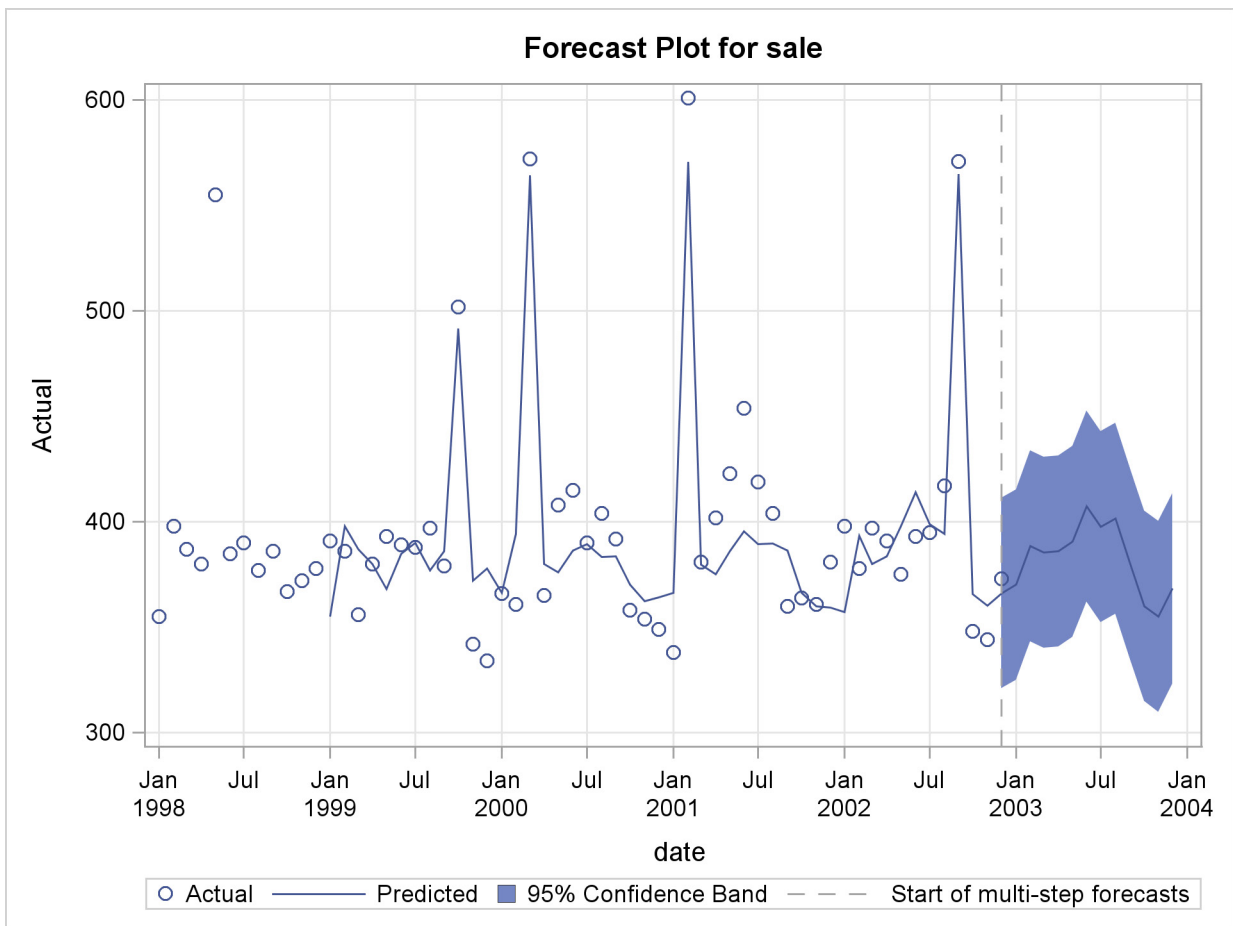
The HPFENGINE procedure then creates a forecast score file using the controllable input variable price, as follows. The designation of this input as “controllable” means that the input may be manipulated in the HPFSCSUB function.

The parameter estimates of the model are shown in [Output 4.4.1](#). A plot of the forecast results is produced and shown in [Figure 4.4.2](#).

```
proc hpfengine data=pricedata
  repository=work.repository
  scorerepository=work.repository
  globalselection=select
  print=estimates
  plot=forecasts;
  id date interval=month;
  forecast sale;
  controllable price / extend=last;
  score;
run;
```

**Output 4.4.1** Parameter Estimates of Selected Model

The HPFENGINE Procedure					
Parameter Estimates for ARIMAX Model					
Component	Parameter	Estimate	Standard Error	t Value	Approx Pr >  t
sale	MA1_12	0.68351	0.13905	4.92	<.0001
price	SCALE	-23.82518	1.34147	-17.76	<.0001

**Output 4.4.2** Forecasts

The NLP procedure is used as follows to maximize an objective function whose definition depends on forecasts from the score file. Some lower bounds are set on the price for certain months.

```
filename score catalog "work.repository.scor0.xml";
proc nlp tech=nmsimp noprint out=outnlp(keep=p1-p6);
  max trev;
  parms p1-p6;
  bounds p3 p4 >= 20.0;
```

```

bounds p1 p2 p5 p6 >= 52.3;
initial = 52.3;
q1 = .; q2 = .; q3 = .;
q4 = .; q5 = .; q6 = .;
call HPFSCSUB('score', 6, 'price', p1, p2, p3, p4, p5, p6,
              'predict', q1, q2, q3, q4, q5, q6);
trev = q1*p1 + q2*p2 + q3*p3 + q4*p4 + q5*p5 +q6*p6;
run;

```

Some manipulation of the output from the NLP procedure is needed before the HPFENGINE procedure is called again with the new future price values. In the following statements, the new price data are transposed and a date variable is added. The result is shown in [Output 4.4.3](#).

```

proc transpose data=outnlp out=outnlp;
  var p1-p6;
run;

data outnlp;
  drop _name_;
  format date date9.;
  set outnlp(rename=(coll=price));
  date = intnx('month', '01dec02'd, _n_);
run;

proc print data=outnlp noobs;
run;

```

**Output 4.4.3** Future Values of Price

	date	price
	01JAN2003	52.3000
	01FEB2003	52.3000
	01MAR2003	34.2393
	01APR2003	34.2512
	01MAY2003	52.3000
	01JUN2003	52.3000

The original data set is now extended, with proper date information, and merged with the results of the optimization, as follows.

```

data pricedataExtend;
  set pricedata;
  drop i;
  output;
  if _n_ ge 60 then do;
    sale = .;
    do i=1 to 6;
      date = intnx('month', '01dec02'd, i);
      output;
    end;
  end;
end;

```

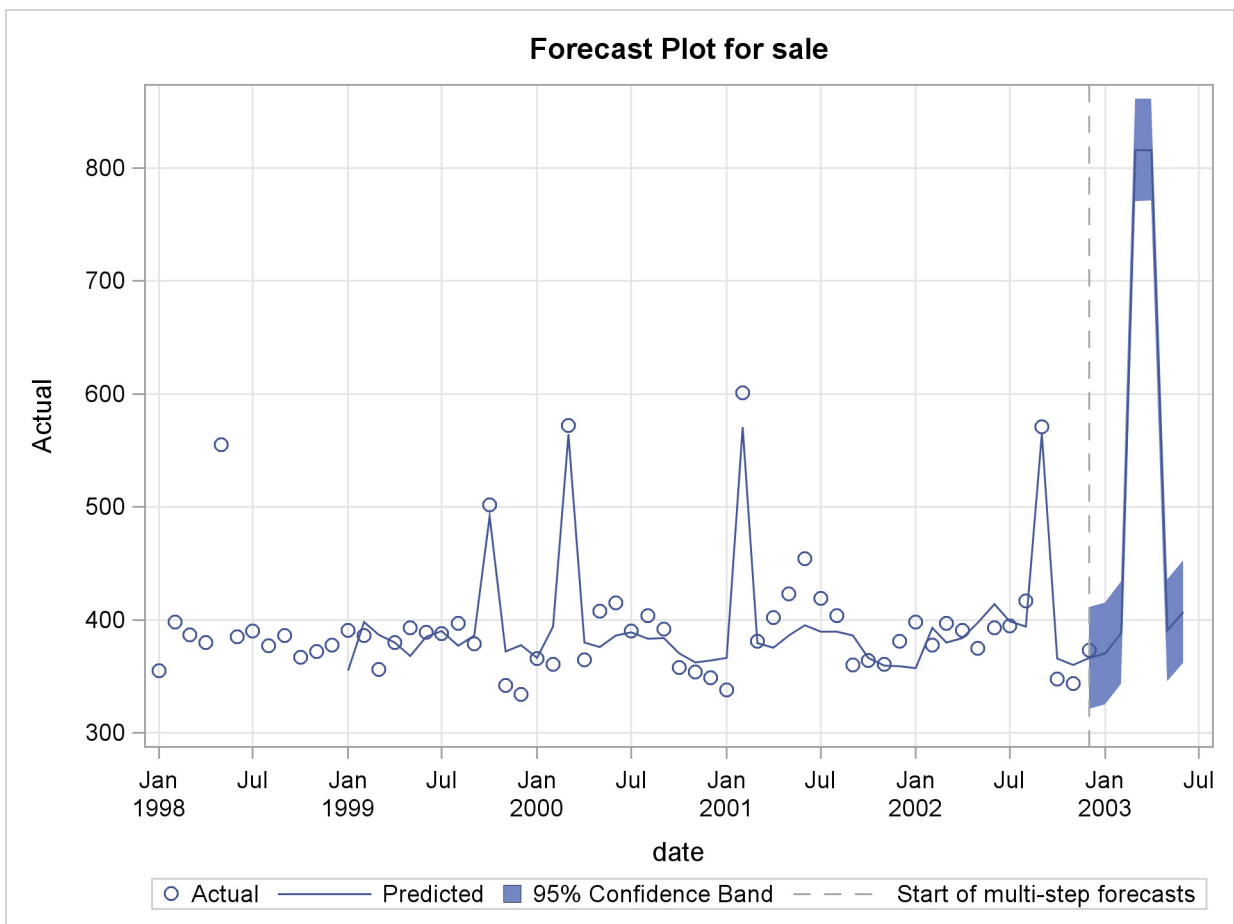
```
run;

data pricedataExtend;
  merge pricedataExtend outnlp;
  by date;
run;
```

The HPFENGINE procedure then uses the optimized price input to forecast future sales, as follows. A plot of the forecasts is shown in [Figure 4.4.4](#).

```
proc hpfengine data=pricedataExtend
  repository=work.repository
  globalselection=select
  plot=forecasts;
  id date interval=month;
  forecast sale;
  input price;
run;
```

**Output 4.4.4** Forecasts



## Example 4.5: HPFENGINE and HPFDIAGNOSE Procedures

The HPFDIAGNOSE procedure is often used in conjunction with the HPFENGINE procedure. This example demonstrates the most basic interaction between the two. In the following statements, model specifications are created by the HPFDIAGNOSE procedure and are those specifications are then fit to the data using the HPFENGINE procedure. The HPFENGINE procedure output is shown in [Output 4.5.1](#). Forecasts are shown in [Figure 4.5.2](#).

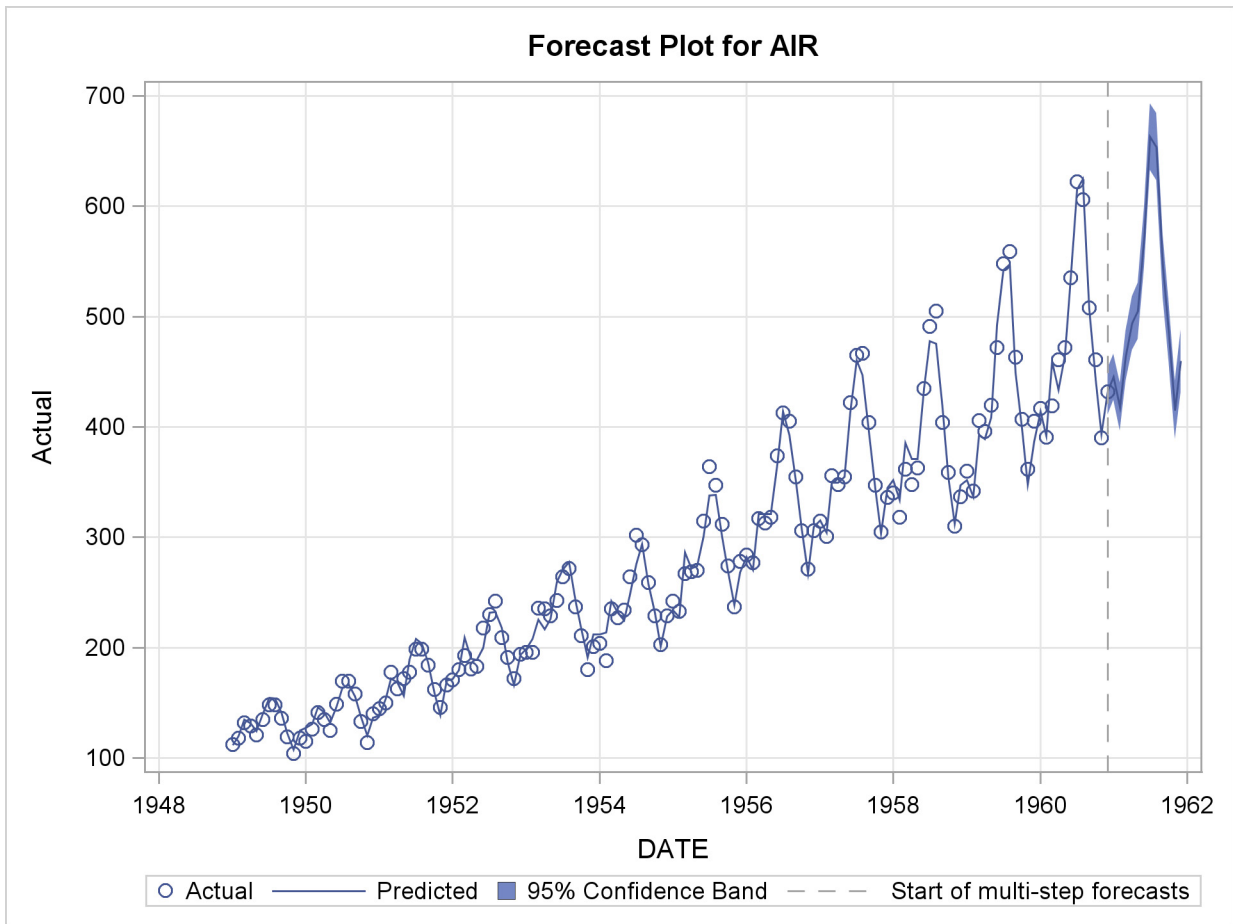
```
proc hpfdiagnose data=sashelp.air
    repository=work.repository
    outest=est;
    id date interval=month;
    forecast air;
run;

proc hpfengine data=sashelp.air
    inest=est outest=outest
    repository=work.repository
    print=(select estimates summary)
    plot=forecasts;
    id date interval=month;
    forecast air;
run;
```

### Output 4.5.1 Selection and Forecast Results

The HPFENGINE Procedure				
Model Selection Criterion = RMSE				
Model	Statistic	Selected	Label	
diag0	10.695241	No	ARIMA: AIR ~ P = 1 D = (1,12)	NOINT
diag1	10.579085	Yes	Winters Method (Multiplicative)	

## Output 4.5.2 Forecasts



It is also possible to compare the performance of model specifications created by the user with those automatically generated by the HPFDIAGNOSE procedure.

In the following statements, two model specifications are created, together with a selection list to reference those specifications.

```
proc hpfarimaspec repository=work.repository name=myarima;
  forecast symbol=sale transform=log q=(1 12) diflist=(1 12) noint;
run;

proc hpfucmspec repository=work.repository name=myucm;
  forecast symbol=sale transform=log;
  irregular;
  level;
  season length=12;
run;

proc hpfselect repository=work.repository name=select;
  spec myarima myucm;
run;
```

The model selection list is passed to the HPFDIAGNOSE procedure using the INSELECTNAME= option as follows. The new selection list ultimately created by the HPFDIAGNOSE procedure will include all model specifications in the INSELECTNAME= selection list together with automatically generated model specifications.

```
proc hpfdiagnose data=sashelp.air
    inselectname=select
    repository=work.repository
    outest=est
    criterion=mape;
    id date interval=month;
    forecast air;
run;
```

The HPFENGINE procedure selects the best-fitting model as follows. Selection results are shown in Figure 4.5.3, parameter estimates of the selected model are shown in Figure 4.5.4, and the forecast summary is shown in Figure 4.5.5.

```
proc hpfengine data=sashelp.air
    inest=est
    repository=work.repository
    print=(select estimates summary)
    plot=forecasts;
    id date interval=month;
    forecast air;
run;
```

**Output 4.5.3** Model Selection Results

The HPFENGINE Procedure					
Model Selection Criterion = MAPE					
Model	Statistic	Selected	Label		
diag3	3.1212192	No	ARIMA: AIR ~ P = 1 D = (1,12)	NOINT	
diag4	3.0845016	No	Winters Method (Multiplicative)		
MYARIMA	2.9672282	Yes	ARIMA: Log( SALE ) ~ D = (1,12) Q = (1,12)	NOINT	
MYUCM	3.1842031	No	UCM: Log( SALE ) = LEVEL + SEASON + ERROR		

**Output 4.5.4** Parameter Estimates of Selected Model

Parameter Estimates for MYARIMA Model					
Component	Parameter	Estimate	Standard Error	t Value	Approx Pr >  t
AIR	MA1_1	0.24576	0.07364	3.34	0.0011
AIR	MA1_12	0.50641	0.07676	6.60	<.0001

**Output 4.5.5** Forecast Summary

Forecast Summary							
Variable	Value	JAN1961	FEB1961	MAR1961	APR1961	MAY1961	JUN1961
AIR	Predicted	450.3513	425.6158	478.6340	501.0466	512.5109	584.8311

Forecast Summary						
Variable	JUL1961	AUG1961	SEP1961	OCT1961	NOV1961	DEC1961
AIR	674.9025	667.8935	558.3906	499.4696	430.1668	479.4592

**Example 4.6: The ADJUST Statement**

This example illustrates the use of adjustment operations. The following statements create seasonal test data with the in-season data trending downward. When the data are forecast, note that the out-of-season range in the forecast horizon is negative, due to the trend. (See the plot in [Figure 4.6.1](#).)

```

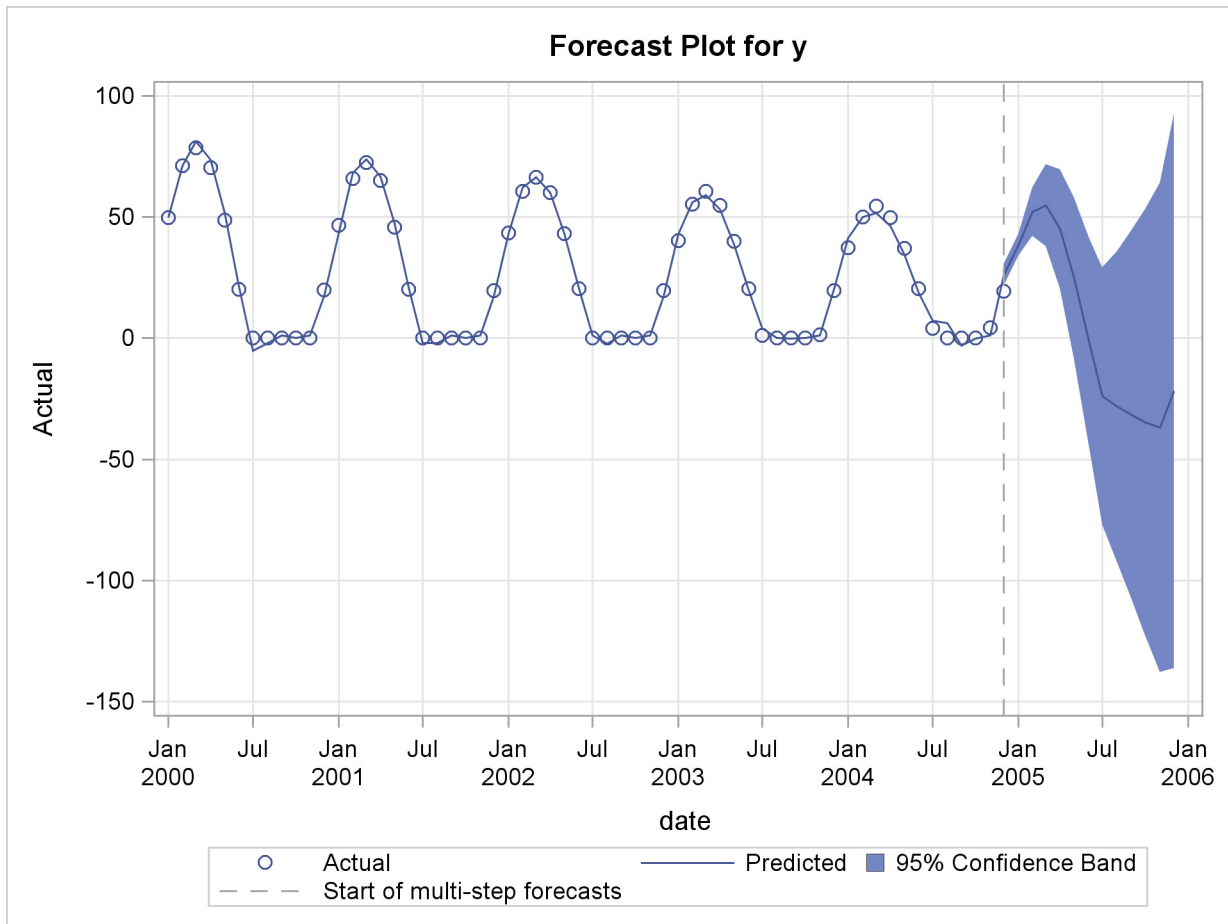
data test;
  do i=1 to 60;
    y = (60-i/2)*sin(2*3.14*i/12) + 20;
    if y lt 0 then do;
      y = 0;
      inseason = 0;
    end;
    else do;
      inseason = 1;
    end;
    date = intnx('month', '01jan2000'd, i-1);
    output;
  end;
  do i=61 to 72;
    y = .;
    date = intnx('month', '01jan2000'd, i-1);
    if i le 66 then inseason = 1;
    else inseason = 0;
    output;
  end;
run;

proc hpfengine data=test
  plot=forecasts;
  id date interval=month;
  forecast y;
run;

```



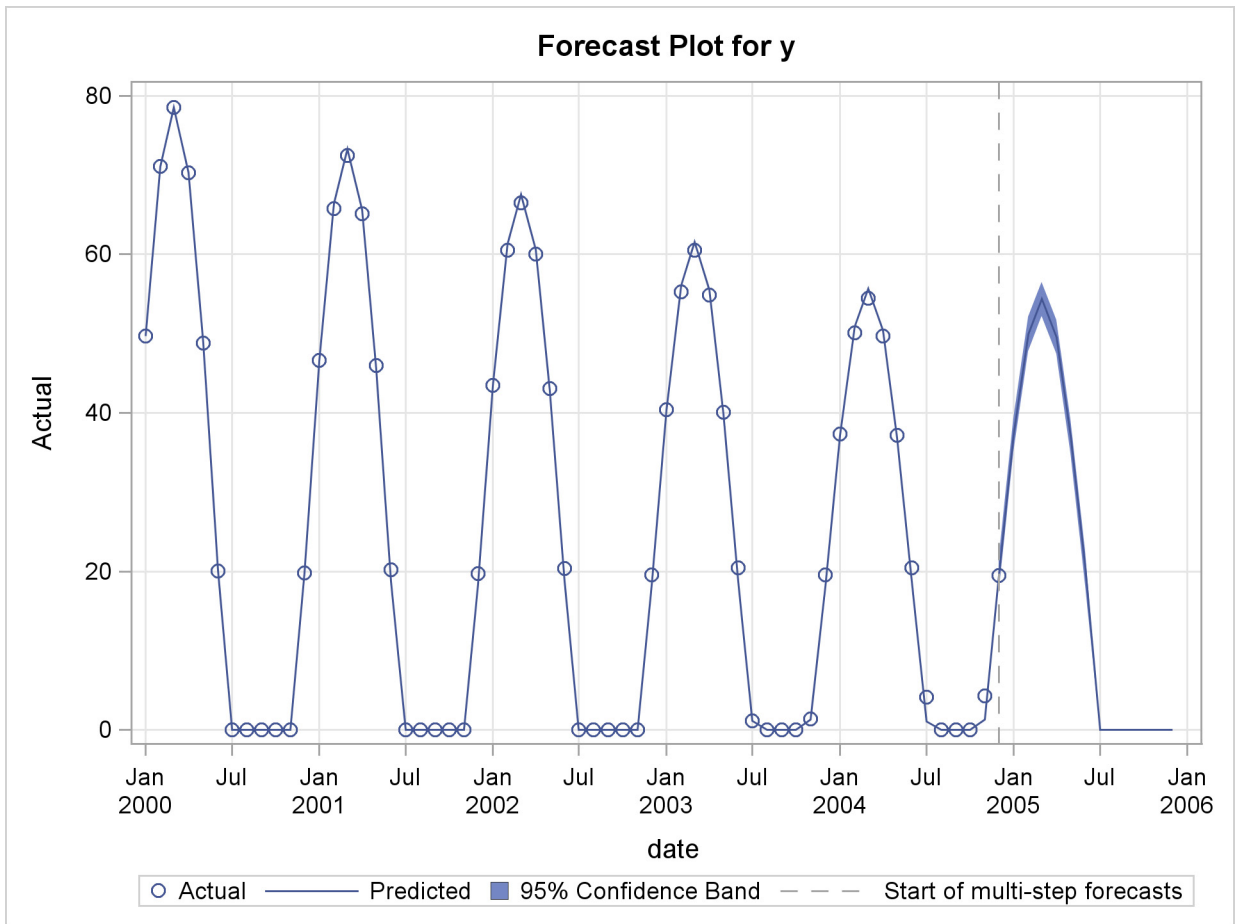
## Output 4.6.1 Forecasts



The seasonal indicator variable `inseason` can be used to fix this situation. Division by zero in the adjustment operation replaces the dividend by a missing value. Subsequent multiplication by zero sets the dividend to zero, especially helpful in the forecast horizon.

Running the `HPFENGINE` procedure again with the adjustment statement, as follows, produces the forecast shown in [Output 4.6.2](#).

```
proc hpfengine data=test
    print=(select estimates)
    plot=forecasts;
    id date interval=month;
    forecast y;
    adjust y=(inseason) / operation=(divide, multiply);
run;
```

**Output 4.6.2** Forecasts**Example 4.7: Multiple Repositories**

This example demonstrates how to use the HPFENGINE procedure to access model specifications and model selection lists in multiple repositories.

In the following statements, three model specifications are created, one in SASUSER.REPOS1 and the other two in SASUSER.REPOS2.

```
proc hpfarimaspec repository=sasuser.repos1 name=myarima;
  forecast symbol=y transform=log q=(1 12) diflist=(1 12) noint;
run;
```

```
proc hpfucmspec repository=sasuser.repos2 name=myucm;
  forecast symbol=y transform=log;
  irregular;
  level;
  slope;
  season length=12;
run;
```

```
proc hpfesmspec repository=sasuser.repos2 name=mywinters;
  esm method=winters transform=log;
run;
```

A selection list is then created using the HPFSELECT procedure and placed in the repository named SASUSER.REPOS3, as follows.

```
proc hpfselect repository=sasuser.repos3 name=mylist;
  spec myarima myucm mywinters;
run;
```

Because the HPFENGINE procedure has only one REPOSITORY option, it would not normally be possible for the procedure to locate the selection list and all three models. The CATNAME statement is used as follows to create a logical concatenation of multiple catalogs; this offers the solution to this problem.

```
catname sasuser.cataloglist (sasuser.repos1 sasuser.repos2 sasuser.repos3);
```

When the HPFENGINE procedure is called, the concatenated name is used in the REPOSITORY= option, as follows. The results of the model selection are shown in [Output 4.7.1](#).

```
proc hpfengine data=sashelp.air
  repository=sasuser.cataloglist
  globalselection=mylist
  print=select;
  id date interval=month;
  forecast air;
run;
```

#### Output 4.7.1 Model Selection Results

The HPFENGINE Procedure			
Model Selection Criterion = MAPE			
Model	Statistic	Selected	Label
MYARIMA	2.9672282	No	ARIMA: $\text{Log}(Y) \sim D = (1,12) \quad Q = (1,12)$ NOINT
MYUCM	3.2601061	No	UCM: $\text{Log}(Y) = \text{TREND} + \text{SEASON} + \text{ERROR}$
MYWINTERS	2.7138783	Yes	Log Winters Method (Multiplicative)

## Example 4.8: ODS Graphics

This example illustrates the use of ODS Graphics. The following statements use the SASHELP.AIR data set to automatically forecast the time series of international airline travel.

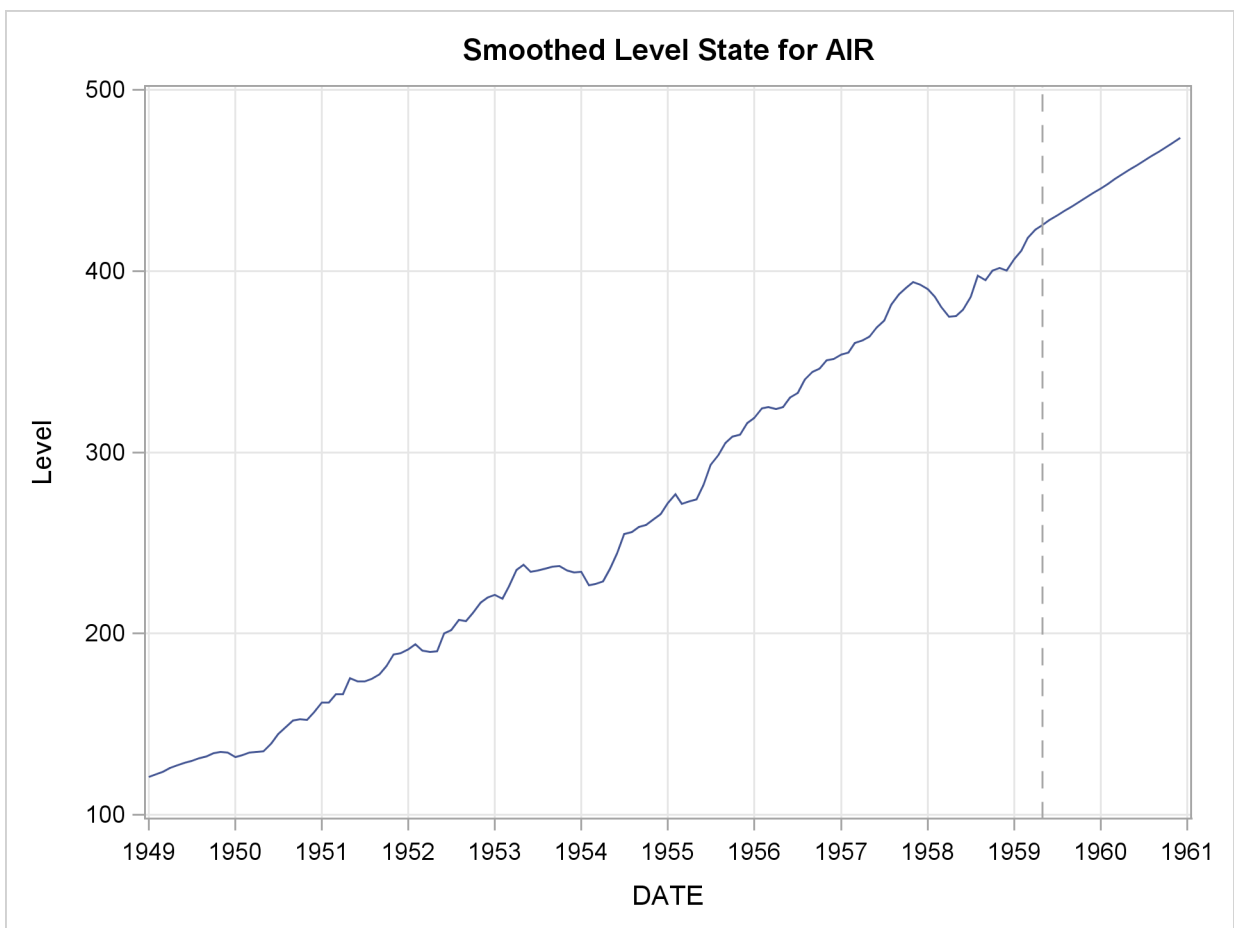
The graphical displays are requested by specifying the ODS GRAPHICS statement and the PLOT= option in the PROC HPFENGINE statement. In this case, all plots are requested. [Figure 4.8.1](#)

through [Figure 4.8.4](#) show a selection of the plots created.

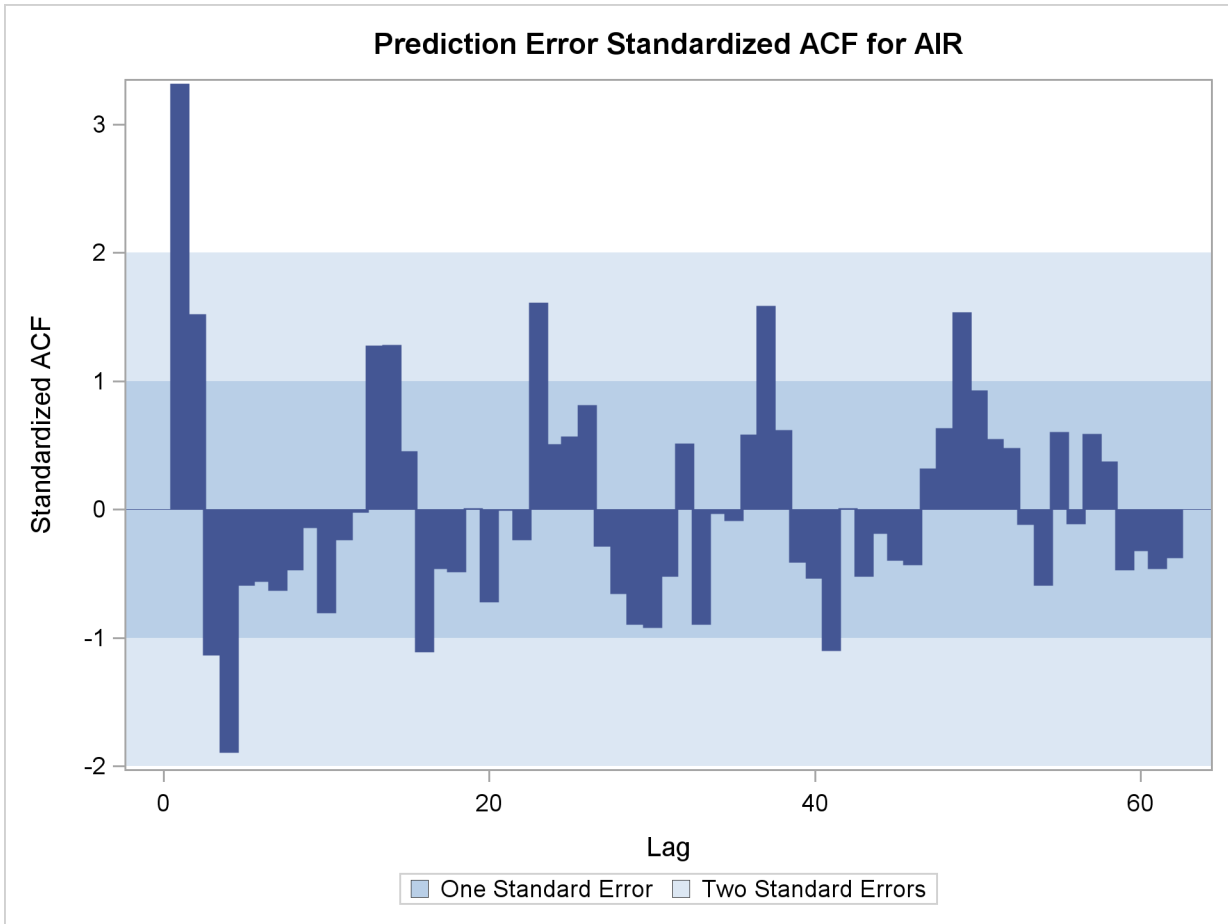
For information about the graphics available in the HPFENGINE procedure, see the “[ODS Graphics](#)” on page 146 section.

```
proc hpfengine data=sashelp.air
    out=_null_
    lead=20
    back=20
    plot=all;
    id date interval=month;
    forecast air;
run;
```

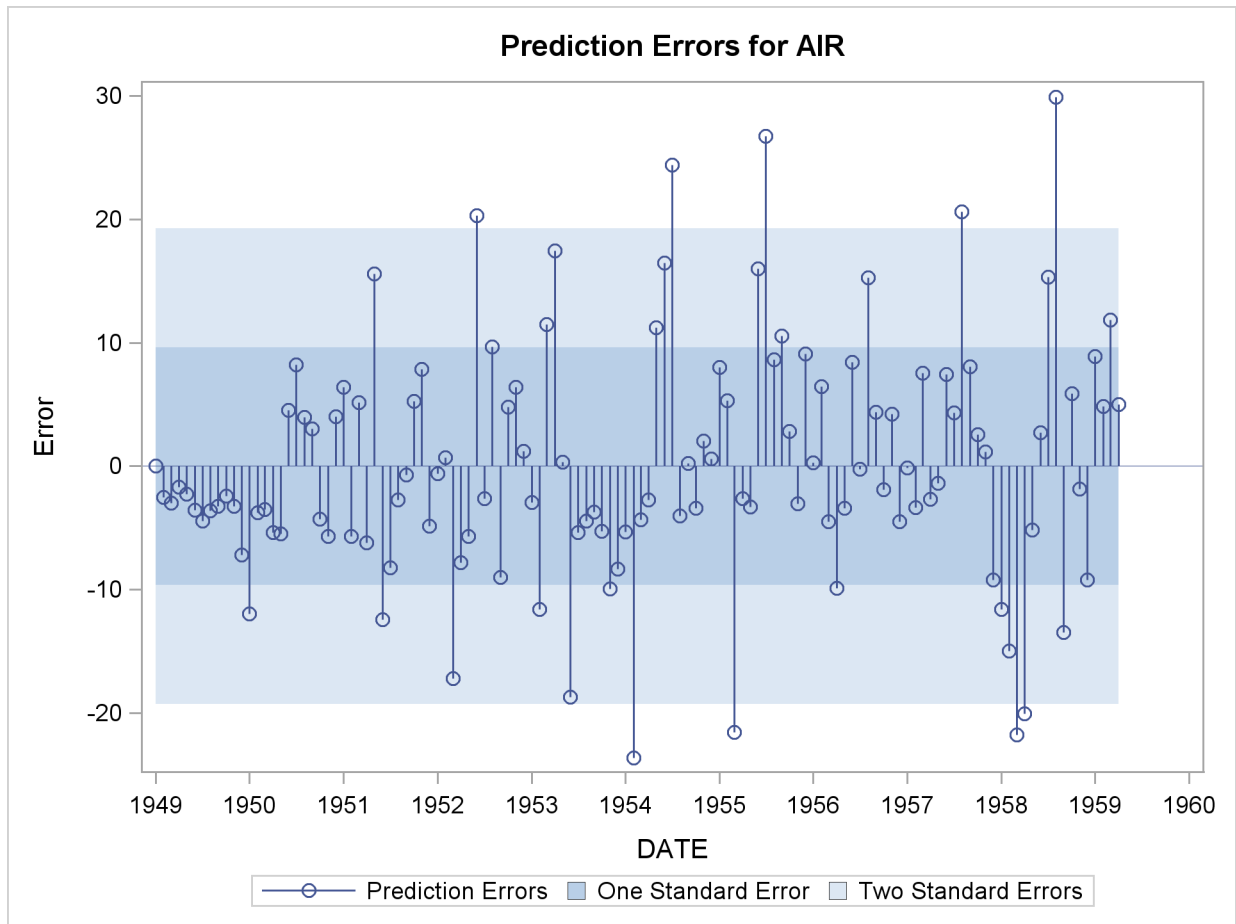
**Output 4.8.1** Component Estimates

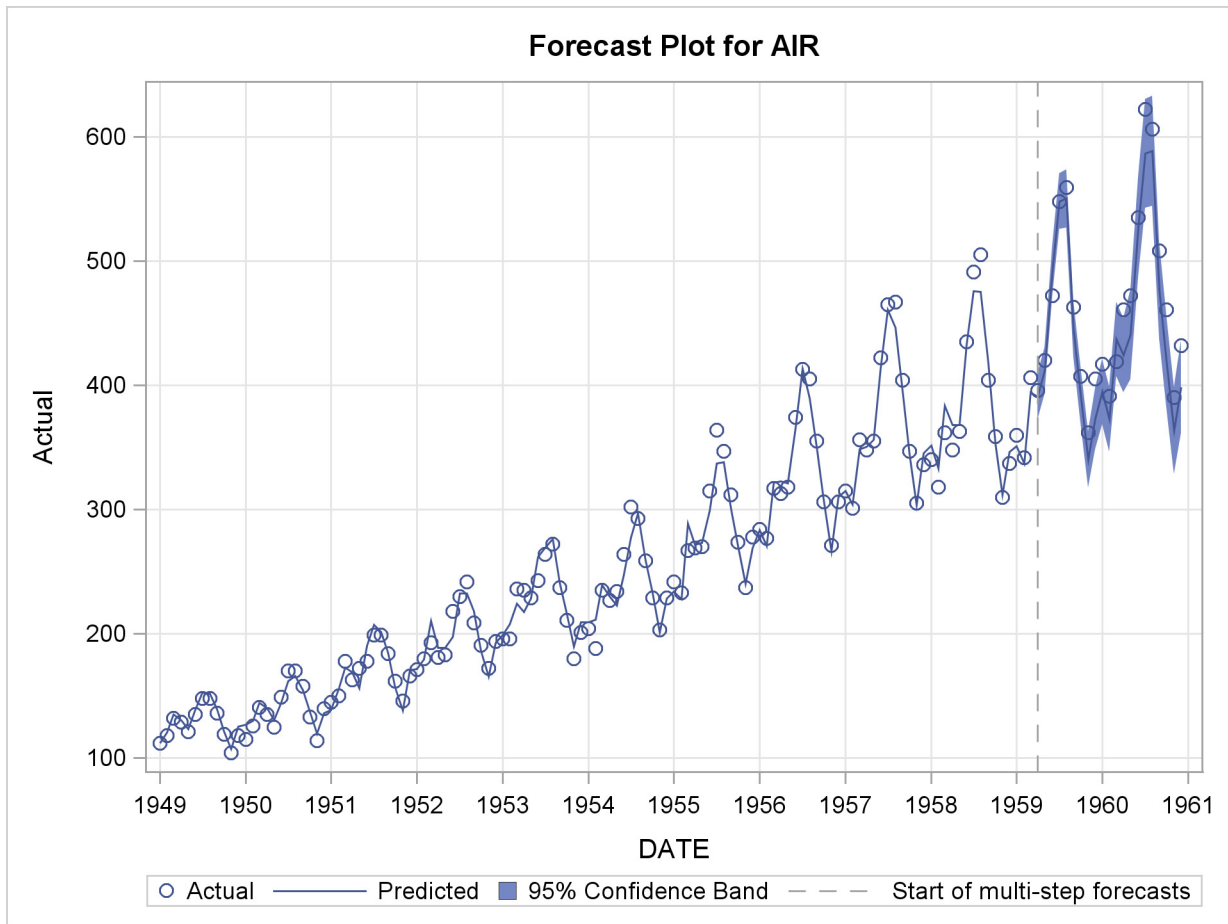


Output 4.8.2 Standardized Autocorrelation of Prediction Errors



Output 4.8.3 Prediction Errors



**Output 4.8.4** Forecasts

---

## References

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994), *Time Series Analysis: Forecasting and Control*, Third Edition, Englewood Cliffs, NJ: Prentice Hall.





## Chapter 5

# The HPFESMSPEC Procedure

### Contents

---

Overview . . . . .	175
Getting Started . . . . .	175
Syntax . . . . .	176
Functional Summary . . . . .	176
PROC HPFESMSPEC Statement . . . . .	177
ESM Statement . . . . .	178
Details . . . . .	181
Smoothing Model Parameter Specification Options . . . . .	181
Examples . . . . .	182
Example 5.1: Various Kinds of ESM Model Specifications . . . . .	182
Example 5.2: Selecting the Best ESM Model . . . . .	185

---

---

## 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
<b>Statements</b>		
specifies the exponential smoothing model	<b>ESM</b>	
<b>Model Repository Options</b>		
specifies the model repository	<b>PROC HPFESMSPEC</b>	<b>REPOSITORY=</b>
specifies the model specification name	<b>PROC HPFESMSPEC</b>	<b>NAME=</b>
specifies the model specification label	<b>PROC HPFESMSPEC</b>	<b>LABEL=</b>
<b>Exponential Smoothing Model Options</b>		
specifies the model selection criterion	<b>ESM</b>	<b>CRITERION=</b>
specifies the damping weight parameter initial value	<b>ESM</b>	<b>DAMPPARM=</b>
specifies the damping weight parameter restrictions	<b>ESM</b>	<b>DAMPREST=</b>
specifies the level weight parameter initial value	<b>ESM</b>	<b>LEVELPARM=</b>
specifies the level weight parameter restrictions	<b>ESM</b>	<b>LEVELREST=</b>
specifies median forecasts	<b>ESM</b>	<b>MEDIAN</b>

Description	Statement	Option
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 season weight parameter initial value	ESM	SEASONPARG=
specifies the season weight parameter restrictions	ESM	SEASONREST=
specifies the time series transformation	ESM	TRANSFORM=
specifies the trend weight parameter initial value	ESM	TRENDPARG=
specifies the trend weight parameter restrictions	ESM	TRENDREST=

## PROC HPFESMSPEC Statement

### PROC HPFESMSPEC *options* ;

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

**LABEL=** *"specification label"*

**SPECLABEL=** *"specification 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*

**SPECNAME=** *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 smoothing model.

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:

#### **CRITERION=** option

#### **SELECT=** option

specifies the model selection criterion (statistic of fit) to be used to select from several candidate models. This option is often used in conjunction with the HOLDOUT= option. The CRITERION= option can also be specified as SELECT=. The default is CRITERION=RMSE.

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
MAPES	Mean Absolute Error Percent of Standard Deviation
MDAPES	Median Absolute Error Percent of Standard Deviation
GMAPES	Geometric Mean Absolute Error Percent of Standard Deviation
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
MASE	Mean Absolute Scaled Error
RSQUARE	R-Square
ADJRSQ	Adjusted R-Square
AADJRSQ	Amemiya's Adjusted R-Square
RWRSQ	Random Walk R-Square
AIC	Akaike Information Criterion
AICC	Finite Sample Corrected AIC
SBC	Schwarz Bayesian Information Criterion
APC	Amemiya's Prediction Criterion

**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.

**LEVELPARG=** *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.

**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 CRITERION= 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
ADDSEASONAL	Additive Seasonal Exponential Smoothing
MULTSEASONAL	Multiplicative Seasonal Exponential Smoothing
SEASONAL	Same as ADDSEASONAL
WINTERS	Winters Multiplicative Method
ADDWINTERS	Winters Additive Method
BEST	Best Candidate Smoothing Model (SIMPLE, DOUBLE, LINEAR, DAMPTREND), (ADDSEASONAL, ADDWINTERS, WINTERS)
BESTN	Best Candidate Nonseasonal Smoothing Model (SIMPLE, DOUBLE, LINEAR, DAMPTREND)
BESTS	Best Candidate Smoothing Model (ADDSEASONAL, ADDWINTERS, WINTERS)

**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.

**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.

**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.

**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( <i>n</i> )	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.

**TRENDPARAM= *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.

## Details

### 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.

---

## Examples

---

### Example 5.1: Various Kinds of ESM Model Specifications

The following example illustrate typical uses of the ESM statement:

```
proc hpfesmspec repository=mymodels
                name=model11
                label="Default Specification";
    esm;
run;

proc hpfesmspec repository=mymodels
                name=model12
                label="Simple Exponential Smoothing";
    esm method=simple;
run;

proc hpfesmspec repository=mymodels
                name=model13
                label="Double Exponential Smoothing";
    esm method=double;
run;

proc hpfesmspec repository=mymodels
                name=model14
                label="Linear Exponential Smoothing";
    esm method=linear;
run;

proc hpfesmspec repository=mymodels
                name=model15
                label="Damp-Trend Exponential Smoothing";
    esm method=damptrend;
run;

proc hpfesmspec repository=mymodels
                name=model16
                label="Seasonal Exponential Smoothing";
    esm method=addseasonal;
run;

proc hpfesmspec repository=mymodels
                name=model17
                label="Winters Method";
    esm method=winters;
run;

proc hpfesmspec repository=mymodels
```



```

        name=model8
        label="Additive-Winters Method";
    esm method=addwinters;
run;

proc hpfesmspec repository=mymodels
    name=model9
    label="Best Smoothing Model";
    esm method=best;
run;

proc hpfesmspec repository=mymodels
    name=model10
    label="Best Non-Seasonal Smoothing Model";
    esm method=bestn;
run;

proc hpfesmspec repository=mymodels
    name=model11
    label="Best Seasonal Smoothing Model";
    esm method=bests;
run;

proc hpfesmspec repository=mymodels
    name=model12
    label="Log Simple Exponential Smoothing";
    esm method=simple transform=log;
run;

proc hpfesmspec repository=mymodels
    name=model13
    label="Log Double Exponential Smoothing";
    esm method=double transform=log;
run;

proc hpfesmspec repository=mymodels
    name=model14
    label="Log Linear Exponential Smoothing";
    esm method=linear transform=log;
run;

proc hpfesmspec repository=mymodels
    name=model15
    label="Log Damp-Trend Exponential Smoothing";
    esm method=damptrend transform=log;
run;

proc hpfesmspec repository=mymodels
    name=model16
    label="Log Seasonal Exponential Smoothing";
    esm method=addseasonal transform=log;
run;

proc hpfesmspec repository=mymodels

```

```
        name=model16
        label="Log Winters Method";
    esm method=winters transform=log;
run;

proc hpfesmspec repository=mymodels
    name=model18
        label="Log Additive-Winters Method";
    esm method=addwinters transform=log;
run;

proc hpfesmspec repository=mymodels
    name=model19
        label="Best Log Smoothing Model";
    esm method=best transform=log;
run;

proc hpfesmspec repository=mymodels
    name=model20
        label="Best Log Non-Seasonal Smoothing Model";
    esm method=bestn transform=log;
run;

proc hpfesmspec repository=mymodels
    name=model21
        label="Best Log Seasonal Smoothing Model";
    esm method=bests transform=log;
run;

title "Models Added to MYMODELS Repository";
proc catalog catalog=mymodels;
    contents;
run;
```

**Output 5.1.1** Listing of Models in MYMODELS repository

Models Added to MYMODELS Repository					
Contents of Catalog WORK.MYMODELS					
#	Name	Type	Create Date	Modified Date	Description
1	MODEL1	XML	09Jul07:16:19:23	09Jul07:16:19:23	Default Specification
2	MODEL10	XML	09Jul07:16:19:24	09Jul07:16:19:24	Best Non-Seasonal Smoothing Model
3	MODEL11	XML	09Jul07:16:19:24	09Jul07:16:19:24	Best Seasonal Smoothing Model
4	MODEL12	XML	09Jul07:16:19:24	09Jul07:16:19:24	Log Simple Exponential Smoothing
5	MODEL13	XML	09Jul07:16:19:24	09Jul07:16:19:24	Log Double Exponential Smoothing
6	MODEL14	XML	09Jul07:16:19:24	09Jul07:16:19:24	Log Linear Exponential Smoothing
7	MODEL15	XML	09Jul07:16:19:24	09Jul07:16:19:24	Log Damp-Trend Exponential Smoothing
8	MODEL16	XML	09Jul07:16:19:24	09Jul07:16:19:24	Log Winters Method
9	MODEL18	XML	09Jul07:16:19:24	09Jul07:16:19:24	Log Additive-Winters Method
10	MODEL19	XML	09Jul07:16:19:24	09Jul07:16:19:24	Best Log Smoothing Model
11	MODEL2	XML	09Jul07:16:19:23	09Jul07:16:19:23	Simple Exponential Smoothing
12	MODEL20	XML	09Jul07:16:19:24	09Jul07:16:19:24	Best Log Non-Seasonal Smoothing Model
13	MODEL21	XML	09Jul07:16:19:24	09Jul07:16:19:24	Best Log Seasonal Smoothing Model
14	MODEL3	XML	09Jul07:16:19:24	09Jul07:16:19:24	Double Exponential Smoothing
15	MODEL4	XML	09Jul07:16:19:24	09Jul07:16:19:24	Linear Exponential Smoothing
16	MODEL5	XML	09Jul07:16:19:24	09Jul07:16:19:24	Damp-Trend Exponential Smoothing
17	MODEL6	XML	09Jul07:16:19:24	09Jul07:16:19:24	Seasonal Exponential Smoothing
18	MODEL7	XML	09Jul07:16:19:24	09Jul07:16:19:24	Winters Method
19	MODEL8	XML	09Jul07:16:19:24	09Jul07:16:19:24	Additive-Winters Method
20	MODEL9	XML	09Jul07:16:19:24	09Jul07:16:19:24	Best Smoothing Model

**Example 5.2: Selecting the Best ESM Model**

This example illustrates how to define models specifications that automatically choose the best exponential smoothing model using MAPE as the model selection criterion.

The following specification 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.

```
proc hpfesmspec repository=mymodels
    name=best_simple;
    esm method=simple transform=auto criterion=mape;
run;
```

The following specification 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.

```
proc hpfesmspec repository=mymodels
    name=best_seasonal;
    esm method=addseasonal transform=auto criterion=mape;
run;
```

The following specification fits 14 forecasting 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.

```
proc hpfesmspec repository=mymodels
    name=best;
    esm method=best transform=auto criterion=mape;
run;
```

## Chapter 6

# The HPFEVENTS Procedure

### Contents

---

Overview: HPFEVENTS Procedure . . . . .	<b>188</b>
Getting Started: HPFEVENTS Procedure . . . . .	<b>189</b>
Syntax: HPFEVENTS Procedure . . . . .	<b>193</b>
Functional Summary . . . . .	194
PROC HPFEVENTS Statement . . . . .	195
BY Statement . . . . .	195
EVENTCOMB Statement . . . . .	195
EVENTDATA Statement . . . . .	196
EVENTDEF Statement . . . . .	196
EVENTDUMMY Statement . . . . .	199
EVENTGROUP Statement . . . . .	199
EVENTKEY Statement . . . . .	201
ID Statement . . . . .	202
VAR Statement . . . . .	203
Details: HPFEVENTS Procedure . . . . .	<b>203</b>
Event Definitions . . . . .	203
Using the EVENTKEY Statement . . . . .	213
Missing Value Interpretation . . . . .	216
Data Set Output . . . . .	216
Printed Output . . . . .	218
Examples: HPFEVENTS Procedure . . . . .	<b>218</b>
Example 6.1: Multiple Timing Values in a Single Event vs. Using Multiple Events and the EVENTCOMB Statement . . . . .	218
Example 6.2: Using a DATA Step to Construct an Events Data Set . . . . .	219
Example 6.3: Preparing a Data Set for PROC HPFENGINE . . . . .	223
Example 6.4: Using SAS Predefined Event Keywords Directly in Other SAS Procedures . . . . .	225
Example 6.5: Viewing Dummy Variables Using SAS/GRAPH Software . . . . .	227
References . . . . .	<b>230</b>

---

---

## Overview: HPFEVENTS Procedure

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.

There are many advantages of using PROC HPFEVENTS:

- 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, such as 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 4, “[The HPFENGINE Procedure](#),” and Chapter 3, “[The HPFDIAGNOSE Procedure](#),” for details.
- SAS predefined EVENT definitions can be accessed directly from PROC HPFENGINE and PROC HPFDIAGNOSE. See the [EVENTKEY](#) statement for a list of SAS predefined EVENT definitions. [Example 6.4](#) illustrates this feature.
- PROC HPFEVENTS can generate a data set that can be used in other procedures such as PROC REG. Refer to Chapter 73, “[The REG Procedure](#)” (*SAS/STAT User’s Guide*). As data are 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: HPFEVENTS Procedure

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 creates events and dummies and outputs the event definitions to a data set named EVDSOUT1 and dummies to a data set named EVDUMOUT1. More examples are shown in the section “[Examples: HPFEVENTS Procedure](#)” on page 218.

```
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 preceding statements are as follows.

- |          |  |
|----------|--|
| 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 <code>_DUR_AFTER_</code> 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 are daily, there are 4 pulses in January of 1950, one on each Saturday. If the data are weekly, a pulse is formed for 4 successive observations in January of 1950. If the data are 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 because garbage is not an event keyword. A warning is printed to the log.

The following statements display the EVENTDATA output data set shown in [Figure 6.1](#).

```
proc print data=evdsout1;
run;
```







Figure 6.3 Event Definition Data Set in Condensed Format

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	.	.	.

Obs	_ENDOBS_	_OBSINTRVL_	_TYPE_	_VALUE_	_PULSE_	_DUR_ BEFORE	_DUR_ 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	.

---

## Syntax: HPFEVENTS Procedure

The following statements are used with the HPFEVENTS procedure:

```

PROC HPFEVENTS < options > ;
  BY variables ;
  EVENTCOMB variable= variable-list / options ;
  EVENTDATA options ;
  EVENTDEF variable= do-list / options ;
  EVENTDUMMY options ;
  EVENTGROUP variable=( list ) ;
  EVENTKEY < variable= > event-keyword < / options > ;
  ID variable INTERVAL= interval options ;
  VAR variables ;

```

## Functional Summary

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

**Table 6.1** Syntax Summary

Description	Statement	Option
<b>Statements</b>		
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	
<b>Data Set Options</b>		
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=
specify format of ID variable	ID	FORMAT=
<b>Dummy Variable Format Options</b>		
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=
<b>Miscellaneous Options</b>		
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 be 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.

---

## 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.

**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.

**RULE=** *value*

specifies the action to take when combining events. The RULE= option accepts the following values: ADD | MAX | MIN | MINNZ | MINMAG | MULT. RULE=ADD is the default.

Table 6.2 explains how the RULE= option is interpreted. Example 6.1 shows how PROC HPFEVENTS interprets the RULE= option in the EVENTDEF and EVENTCOMB statements.

## 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.

**CONDENSE**

specifies that the EVENTDATA OUT= data set be 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” on page 216 section.

**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.

## EVENTDEF Statement

**EVENTDEF** *SAS-variable-name= timing-value-list / qualifier options* ;

The EVENTDEF statement defines an event that may be included in forecasting models. The following options can be used with the EVENTDEF statement.

**AFTER=( < DURATION=value > < SLOPE=value > )**

specifies options that control the event definition after the timing value. The DURATION= and SLOPE= suboptions are used within the parentheses in the AFTER=( ) option. See the BEFORE= option for information on the DURATION= and SLOPE= suboptions.

**BEFORE=( < DURATION=value > < SLOPE=value > )**

specifies options that control the event definition before the timing value. The DURATION= and SLOPE= options are used within the parentheses in the BEFORE=( ) option.

**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.

**SLOPE= GROWTH | DECAY**

specifies whether a ramp or temporary change type is growth or decay. 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. SLOPE= is ignored unless TYPE=RAMP, TYPE=TR, TYPE=TEMPRAMP, or TYPE=TC. SLOPE= is also ignored if the corresponding DURATION=0.

SLOPE=GROWTH is the default in all cases except TYPE=TC. For TYPE=TC, the default is BEFORE=(SLOPE=GROWTH) and AFTER=(SLOPE=DECAY).

The 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.

**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.

**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 Chapter 3, "Date Intervals, Formats, and Functions" (*SAS/ETS User's Guide*), for the intervals that can be specified.

**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 Chapter 3, “Date Intervals, Formats, and Functions” (*SAS/ETS User’s Guide*), for the intervals that can be specified.

**RULE=** *value*

specifies the action to take when the defined event has multiple timing values that 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**. **RULE=ADD** is the default. Table 6.2 explains how the **RULE=** option is interpreted when the value for an observation is defined by multiple timing values.

**Table 6.2** Definition of **RULE=** Option Values

<b>RULE= Option</b>	<b>Name</b>	<b>Definition</b>
<b>ADD</b>	Add	Add the values.
<b>MAX</b>	Maximum	Use the maximum value.
<b>MIN</b>	Minimum	Use the minimum value.
<b>MINNZ</b>	Minimum nonzero	Use the minimum nonzero value.
<b>MINMAG</b>	Minimum magnitude	Use the value whose magnitude is the least.
<b>MULT</b>	Multiply	Multiply the values.

Because the range of the event associated with a timing value might not include all the observations in the series, **RULE=** may be interpreted differently when you are 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: HPFEVENTS Procedure](#)” on page 218, [Example 6.1](#) shows how PROC HPFEVENTS interprets each of these statements.

**SHIFT=** *number*

specifies the number of pulses to shift the timing value,  $\delta$ . 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).

**TCPARM=** *number*

specifies a parameter  $0 \leq \phi \leq 1$  used in the growth/decay equation for TYPE=TC given in Table 6.8. 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.

**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 6.9 and Table 6.10 illustrate the basic shape for each TYPE= value. TYPE=POINT is the default type.

**VALUE=** *number*

specifies the event indicator value,  $\nu$ . The default event indicator value is one ( $\nu = 1.0$ ). Table 6.8 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.

---

## EVENTDUMMY Statement

**EVENTDUMMY** *OUT=SAS-data-set* ;

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.

---

## EVENTGROUP Statement

**EVENTGROUP** *variable= ( variable-list ) < / option >* ;

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, you 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 stored in the EVENTDATA OUT= data set. If no label is specified, then "." is displayed in the EVENTDATA OUT= data set.

[Table 6.3](#) 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 6.3** 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 6.11 gives more detail on the seasonal and trend SAS predefined EVENTS that compose the EVENT groups.

---

## EVENTKEY Statement

**EVENTKEY** < variable=> 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.

See the section “Using the EVENTKEY Statement” on page 213 for more information on the EVENTKEY Statement.

---

## 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 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 Chapter 3, "Date Intervals, Formats, and Functions" (*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 want to 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 contain the same number of observations.

---

## 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: HPFEVENTS Procedure

---

### Event Definitions

This section describes the use of the EVENTDEF statement.

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 following EVENTDEF statement 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 follows: 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 6.4 shows the date keywords that can be used in a timing value list and their definitions.

**Table 6.4** Holiday Date Keywords and Definitions

<b>Date Keyword</b>	<b>Definition</b>
BOXING	December 26th
CANADA	July 1st
CANADAOBSERVED	July 1st, or July 2nd, if July 1st is a Sunday
CHRISTMAS	December 25th
COLUMBUS	2nd Monday in October
EASTER	Easter Sunday
FATHERS	3rd Sunday in June
HALLOWEEN	October 31st
LABOR	1st Monday in September
MLK	3rd Monday in January
MEMORIAL	last Monday in May
MOTHERS	2nd Sunday in May
NEWYEAR	January 1st
THANKSGIVING	4th Thursday in November
THANKSGIVINGCANADA	2nd Monday in October
USINDEPENDENCE	July 4th
USPRESIDENTS	3rd Monday in February (since 1971)
VALENTINES	February 14th
VETERANS	November 11th
VETERANSUSG	U.S. government observed date for Monday-Friday schedule
VETERANSUSPS	U.S. government observed date for Monday-Saturday schedule (U.S. Post Office)
VICTORIA	Monday on or preceding May 24th

The date of Easter is calculated using a method described by Montes (2001b).

Table 6.5 shows the seasonal date keywords that can be used in a timing value list and their definitions.

**Table 6.5** 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 $n$ th 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.

## Event Types

The TYPE= option in the EVENTDEF statement specifies the type of the event defined. These event types are POINT, LS, RAMP, TR or TEMPRAMP, TC, LIN or LINEAR, QUAD, CUBIC, INV or INVERSE, and LOG or LOGARITHMIC.

Table 6.6, Table 6.7, and Table 6.8 show the formulas used to calculate the dummy variables for the event. Table 6.7 shows the formula used for each type of event when the event extends infinitely both before and after the timing value. Table 6.8 shows the formula used for each type of event for finite duration values.

To calculate the dummy values, the timing values list is first expanded, if applicable, with respect to the PERIOD option and the time series ID values. Then, if SHIFT is not zero, the timing values are shifted according to the value of SHIFT and PULSE. In the formulas,  $t_i$  is the observation specified by the  $i$ th (shifted) timing value in the expanded timing value list, VALUE= $v$ , TCPARM= $\phi$ , AFTER=(DURATION= $n$  SLOPE= $s_a$ ), BEFORE=(DURATION= $m$  SLOPE= $s_b$ ), and PULSE= $interval$ . Table 6.6 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 beyond the top of the ramp, either before  $t_b$  or after  $t_e$ .) 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 evaluating multiple timing values, due to the user either specifying multiple values or specifying PERIOD=, observations within the range of the function will be calculated according to the



formulas in the tables. Observations not in the range of the functions will be left undefined. When the range from two timing values overlap, the **RULE= option** applies. After all timing values have been evaluated, the undefined values will be set to zero.

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: HPFEVENTS Procedure](#)” on page 218, [Example 6.5](#) shows how PROC HPFEVENTS interprets each of these statements.

**Table 6.6** Calculating the Beginning and Ending Observation for Events

<b>BEFORE=</b> <b>(DURATION=value)</b>	<b>PULSE=value</b>	<b>Definition of <math>t_b</math></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 shifted timing value
$m > 0$	not specified	$t_b = t_i - m$
$m \geq 0$	<i>interval</i>	$t_b =$ the observation specified by the date INTNX( <i>interval</i> , timing value, $-m$ , ‘begin’ )
<b>AFTER=</b> <b>(DURATION=value)</b>	<b>PULSE=value</b>	<b>Definition of <math>t_e</math></b>
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 shifted timing value
$n > 0$	not specified	$t_e = t_i + n$
$n \geq 0$	<i>interval</i>	$t_e =$ the observation specified by the date INTNX( <i>interval</i> , timing value, $n$ , ‘end’ )

**Table 6.7** Event Types for Infinite Durations ( $m = \text{ALL}$  and  $n = \text{ALL}$ )

<b>Type</b>	<b>Description</b>	<b>Definition</b>
POINT	point or pulse	$\xi_{it} = v$ , for all $t$

Table 6.7 continued

Type	Description	Definition
LS	level shift	$\xi_{it} = v$ , for all $t$
RAMP	ramp SLOPE=GROWTH	$\xi_{it} = v(t - t_i)$ , for all $t$
	SLOPE=DECAY	$\xi_{it} = v(t_i - t)$ , for all $t$
	$s_b = \text{GROWTH}$ $s_a = \text{DECAY}$	$\xi_{it} = v(t - t_i)$ , if $t \leq t_i$ $\xi_{it} = v(t_i - t)$ , if $t_i \leq t$
	$s_b = \text{DECAY}$ $s_a = \text{GROWTH}$	$\xi_{it} = v(t_i - t)$ , if $t \leq t_i$ $\xi_{it} = v(t - t_i)$ , if $t_i \leq t$
TEMPRAMP or TR	temporary ramp	TEMPRAMP is the same as RAMP for infinite cases
TC	temporary change SLOPE=GROWTH	$\xi_{it} = v\phi^{(t_i-t)}$ , for all $t$
	SLOPE=DECAY	$\xi_{it} = v\phi^{(t-t_i)}$ , for all $t$
	$s_b = \text{GROWTH}$ $s_a = \text{DECAY}$	$\xi_{it} = v\phi^{(t_i-t)}$ , if $t \leq t_i$ $\xi_{it} = v\phi^{(t-t_i)}$ , if $t_i \leq t$
	$s_b = \text{DECAY}$ $s_a = \text{GROWTH}$	$\xi_{it} = v\phi^{(t-t_i)}$ , if $t \leq t_i$ $\xi_{it} = v\phi^{(t_i-t)}$ , if $t_i \leq t$
LINEAR or LIN	linear trend SLOPE= does not apply	$\xi_{it} = v(t - t_i)$ , for all $t$
QUAD	quadratic trend SLOPE= does not apply	$\xi_{it} = v(t - t_i)^2$ , for all $t$
CUBIC	cubic trend SLOPE= does not apply	$\xi_{it} = v(t - t_i)^3$ , for all $t$
INVERSE or INV	inverse trend SLOPE= does not apply	$\xi_{it} = v/t$ , for all $t$
LOGARITHMIC	log trend	$\xi_{it} = v \log(t)$ , for all $t$

Table 6.7 continued

Type	Description	Definition
or LOG	SLOPE= does not apply	

Table 6.8 Event Types (for Finite  $m, n$ )

Type	Description	Definition
POINT	point or pulse	$\xi_{it} = v$ , if $t_b \leq t \leq t_e$ $\xi_{it} = \text{undefined}$ , otherwise
LS	level shift	$\xi_{it} = v$ , if $t_b \leq t \leq t_e$ $\xi_{it} = \text{undefined}$ , otherwise
RAMP	ramp $m = n = 0$ PULSE= $interval \leq$ width of an observation	$\xi_{it} = 0$ , if $t = t_i$ $\xi_{it} = \text{undefined}$ , otherwise
	SLOPE=GROWTH	$\xi_{it} = v(t - t_b)/(t_e - t_b)$ , if $t_b \leq t \leq t_e$ $\xi_{it} = v$ , if $t > t_e$ $\xi_{it} = \text{undefined}$ , otherwise
	SLOPE=DECAY	$\xi_{it} = v$ , if $t < t_b$ $\xi_{it} = v(t_e - t)/(t_e - t_b)$ , if $t_b \leq t \leq t_e$ $\xi_{it} = \text{undefined}$ , otherwise
	$s_b = \text{GROWTH}$ $s_a = \text{DECAY}$ $m > 0, n > 0$	$\xi_{it} = v(t - t_b)/(t_i - t_b)$ , if $t_b \leq t \leq t_i$ $\xi_{it} = v(t_e - t)/(t_e - t_i)$ , if $t_i \leq t \leq t_e$ $\xi_{it} = \text{undefined}$ , otherwise
	$s_b = \text{DECAY}$ $s_a = \text{GROWTH}$ $m > 0, n > 0$	$\xi_{it} = v$ , if $t < t_b$ $\xi_{it} = v(t_i - t)/(t_i - t_b)$ , if $t_b \leq t \leq t_i$ $\xi_{it} = v(t - t_i)/(t_e - t_i)$ , if $t_i \leq t \leq t_e$ $\xi_{it} = v$ , if $t > t_e$
TEMPRAMP or TR	temporary ramp $m = n = 0$ PULSE= $interval \leq$ width of an observation	$\xi_{it} = 0$ , if $t = t_i$ $\xi_{it} = \text{undefined}$ , otherwise
	SLOPE=GROWTH	$\xi_{it} = v(t - t_b)/(t_e - t_b)$ , if $t_b \leq t \leq t_e$ $\xi_{it} = \text{undefined}$ , otherwise
	SLOPE=DECAY	$\xi_{it} = v(t_e - t)/(t_e - t_b)$ , if $t_b \leq t \leq t_e$ $\xi_{it} = \text{undefined}$ , otherwise

Table 6.8 continued

Type	Description	Definition
	$s_b = \text{GROWTH}$ $s_a = \text{DECAY}$ $m > 0, n > 0$	$\xi_{it} = v(t - t_b)/(t_i - t_b)$ , if $t_b \leq t \leq t_i$ $\xi_{it} = v(t_e - t)/(t_e - t_i)$ , if $t_i \leq t \leq t_e$ $\xi_{it} = \text{undefined}$ , otherwise
	$s_b = \text{DECAY}$ $s_a = \text{GROWTH}$ $m > 0, n > 0$	$\xi_{it} = v(t_i - t)/(t_i - t_b)$ , if $t_b \leq t \leq t_i$ $\xi_{it} = v(t - t_i)/(t_e - t_i)$ , if $t_i \leq t \leq t_e$ $\xi_{it} = \text{undefined}$ , otherwise
TC	temporary change SLOPE=GROWTH	$\xi_{it} = v\phi^{(t_e-t)}$ , if $t_b \leq t \leq t_e$ $\xi_{it} = \text{undefined}$ , otherwise
	SLOPE=DECAY	$\xi_{it} = v\phi^{(t-t_b)}$ , if $t_b \leq t \leq t_e$ $\xi_{it} = \text{undefined}$ , otherwise
	$s_b = \text{GROWTH}$ $s_a = \text{DECAY}$ $m > 0, n > 0$	$\xi_{it} = v\phi^{(t_i-t)}$ , if $t_b \leq t \leq t_i$ $\xi_{it} = v\phi^{(t-t_i)}$ , if $t_i \leq t \leq t_e$ $\xi_{it} = \text{undefined}$ , otherwise
	$s_b = \text{DECAY}$ $s_a = \text{GROWTH}$ $0 < m \leq n$	$\xi_{it} = v\phi^{((t_e-t_i)+(t-t_i))}$ , if $t_b \leq t \leq t_i$ $\xi_{it} = v\phi^{(t_e-t)}$ , if $t_i \leq t \leq t_e$ $\xi_{it} = \text{undefined}$ , otherwise
	$s_b = \text{DECAY}$ $s_a = \text{GROWTH}$ $0 < n \leq m$	$\xi_{it} = v\phi^{(t-t_b)}$ , if $t_b \leq t \leq t_i$ $\xi_{it} = v\phi^{((t_i-t_b)+(t-t_i))}$ , if $t_i \leq t \leq t_e$ $\xi_{it} = \text{undefined}$ , otherwise
LINEAR or LIN	linear trend SLOPE= does not apply	$\xi_{it} = v(t - t_i)$ , if $t_b \leq t \leq t_e$
QUAD	quadratic trend SLOPE= does not apply	$\xi_{it} = v(t - t_i)^2$ , if $t_b \leq t \leq t_e$
CUBIC	cubic trend SLOPE= does not apply	$\xi_{it} = v(t - t_i)^3$ , if $t_b \leq t \leq t_e$
INVERSE or INV	inverse trend SLOPE= does not apply	$\xi_{it} = v/(t - t_b + 1)$ , if $t_b \leq t \leq t_e$
LOGARITHMIC	log trend	$\xi_{it} = v \log(t - t_b + 1)$ , if $t_b \leq t \leq t_e$

**Table 6.8** *continued*

Type	Description	Definition
or LOG	SLOPE= does not apply	

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.

## Details of Event Specifications

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:

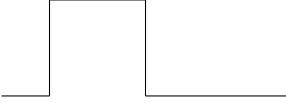



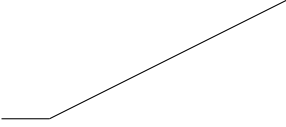
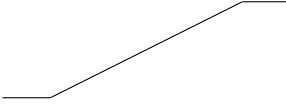
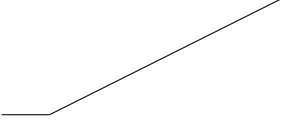
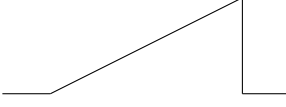




```
EVENTDEF E1= '01JAN1950'D / BEFORE=(DURATION=3)
              AFTER=(DURATION=4);
```

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 6.9](#) and [Table 6.10](#) show 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*.

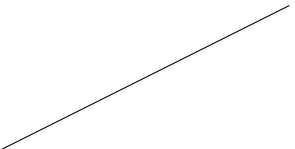
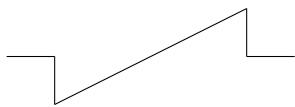
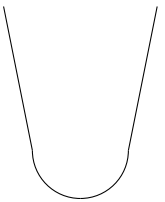

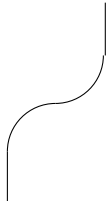
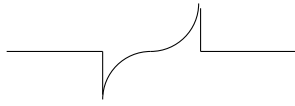
**Table 6.9** Default DURATION= Values for Non-trend Types

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)		
TYPE=INV or INVERSE	default is ALL (or end of series)		

**Table 6.9** *continued*

Non-trend TYPE= (BEFORE= Default Is 0)	AFTER= (DURATION=) Default	Default Shape	Shape When Finite AFTER= Duration > 0
TYPE=LOG or LOGARITHMIC	default is ALL (or end of series)		

**Table 6.10** Default DURATION= Values for Trend Types

Trend TYPE= 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)		

## Using the EVENTKEY Statement

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 section “Getting Started: HPFEVENTS Procedure” on page 189, the default SAS variable name for the predefined event is the predefined event keyword. However, the user can specify a SAS 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, a trend keyword, or an additive outlier or level shift based on a timing value. Table 6.11 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 preceding events are stored in a data set named SPRINGHOLIDAYS. The first EVENTKEY statement in the following example 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 6.11** 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 6.4](#) that can be used in the `EVENTDEF` statement can be used as SAS predefined event keywords. The timing value(s) are as defined in [Table 6.4](#) and the default qualifiers are as shown in [Table 6.11](#). [Table 6.5](#) 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 6.11](#). [Table 6.12](#) gives a more detailed description of how date and observation numbers are encoded into AO and LS type predefined events.

**Table 6.12** 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

## 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 4, “The HPFENGINE Procedure,” for more details.) 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 are 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.

<code>_CLASS_</code>	Class of EVENT: SIMPLE, COMBINATION, PREDEFINED. The default for <code>_CLASS_</code> is SIMPLE.
<code>_KEYNAME_</code>	Contains either a date keyword (SIMPLE EVENT) or a predefined EVENT variable name (PREDEFINED EVENT) or an event name (COMBINATION event). All <code>_KEYNAME_</code> values are displayed in uppercase. However, if the <code>_KEYNAME_</code> value refers to an event name, then the actual name may be mixed case. The default for <code>_KEYNAME_</code> is no keyname, designated by “.”.
<code>_STARTDATE_</code>	Contains either the date timing value or the first date timing value to use in a do-list. The default for <code>_STARTDATE_</code> is no date, designated by a missing value.
<code>_ENDDATE_</code>	Contains the last date timing value to use in a do-list. The default for <code>_ENDDATE_</code> is no date, designated by a missing value.
<code>_DATEINTRVL_</code>	Contains the interval for the date do-list. The default for <code>_DATEINTRVL_</code> is no interval, designated by “.”.
<code>_STARTDT_</code>	Contains either the datetime timing value or the first datetime timing value to use in a do-list. The default for <code>_STARTDT_</code> is no datetime, designated by a missing value.
<code>_ENDDT_</code>	Contains the last datetime timing value to use in a do-list. The default for <code>_ENDDT_</code> is no datetime, designated by a missing value.
<code>_DTINTRVL_</code>	Contains the interval for the datetime do-list. The default for <code>_DTINTRVL_</code> is no interval, designated by “.”.
<code>_STARTOBS_</code>	Contains either the observation number timing value or the first observation number timing value to use in a do-list. The default for <code>_STARTOBS_</code> is no observation number, designated by a missing value.
<code>_ENDOBS_</code>	Contains the last observation number timing value to use in a do-list. The default for <code>_ENDOBS_</code> is no observation number, designated by a missing value.
<code>_OBSINTRVL_</code>	Contains the interval length of the observation number do-list. The default for <code>_OBSINTRVL_</code> is no interval, designated by “.”.
<code>_TYPE_</code>	Type of EVENT. The default for <code>_TYPE_</code> is POINT.
<code>_VALUE_</code>	Value for nonzero observation. The default for <code>_VALUE_</code> is 1.0.
<code>_PULSE_</code>	INTERVAL that defines the units for the DURATION values. The default for <code>_PULSE_</code> is no interval (one observation), designated by “.”.
<code>_DUR_BEFORE_</code>	Number of durations before the timing value. The default for <code>_DUR_BEFORE_</code> is 0.
<code>_DUR_AFTER_</code>	Number of durations after the timing value. The default for <code>_DUR_AFTER_</code> is 0.
<code>_SLOPE_BEFORE_</code>	For TYPE=RAMP, TYPE=RAMPP, and TYPE=TC, this determines whether the curve is GROWTH or DECAY before the timing value. The default for <code>_SLOPE_BEFORE_</code> is GROWTH.
<code>_SLOPE_AFTER_</code>	For TYPE=RAMP, TYPE=RAMPP, and TYPE=TC, this determines whether the curve is GROWTH or DECAY after the timing value. The default for <code>_SLOPE_AFTER_</code> is GROWTH unless TYPE=TC, then the default is DECAY.

<code>_SHIFT_</code>	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 <code>_SHIFT_</code> is 0.
<code>_TCPARM_</code>	Parameter for EVENT of TYPE=TC. The default for <code>_TCPARM_</code> is 0.5.
<code>_RULE_</code>	Rule to use when combining events or when timing values of an event overlap. The default for <code>_RULE_</code> is ADD.
<code>_PERIOD_</code>	Frequency interval at which the event should be repeated. If this value is missing, then the event is not periodic. The default for <code>_PERIOD_</code> is no interval, designated by “.”.
<code>_LABEL_</code>	Label or description for the event. If the user does not specify a label, then the default label value will be displayed as “.”. However, the default label used in an EVENTDUMMY OUT= data set is “Dummy Variable for Event <variable-name>”. See the <a href="#">LABEL= option</a> 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: HPFEVENTS Procedure

---

### Example 6.1: 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.

```

data a(Label='Box-Jenkins Series G: International Airline Data');
  set sashelp.air;
  t = intnx( 'month', '01jan1949'd, _n_-1 );
  format t DATE.;
run;

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 6.1.1** 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 zeros. 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 6.2: Using a DATA Step to Construct an Events Data Set

This 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 statements construct 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(obs=10);
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 6.2.1** 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 6.2.2** 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

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 SGPLOT to visually verify that these potential outliers are appropriate for the original data.

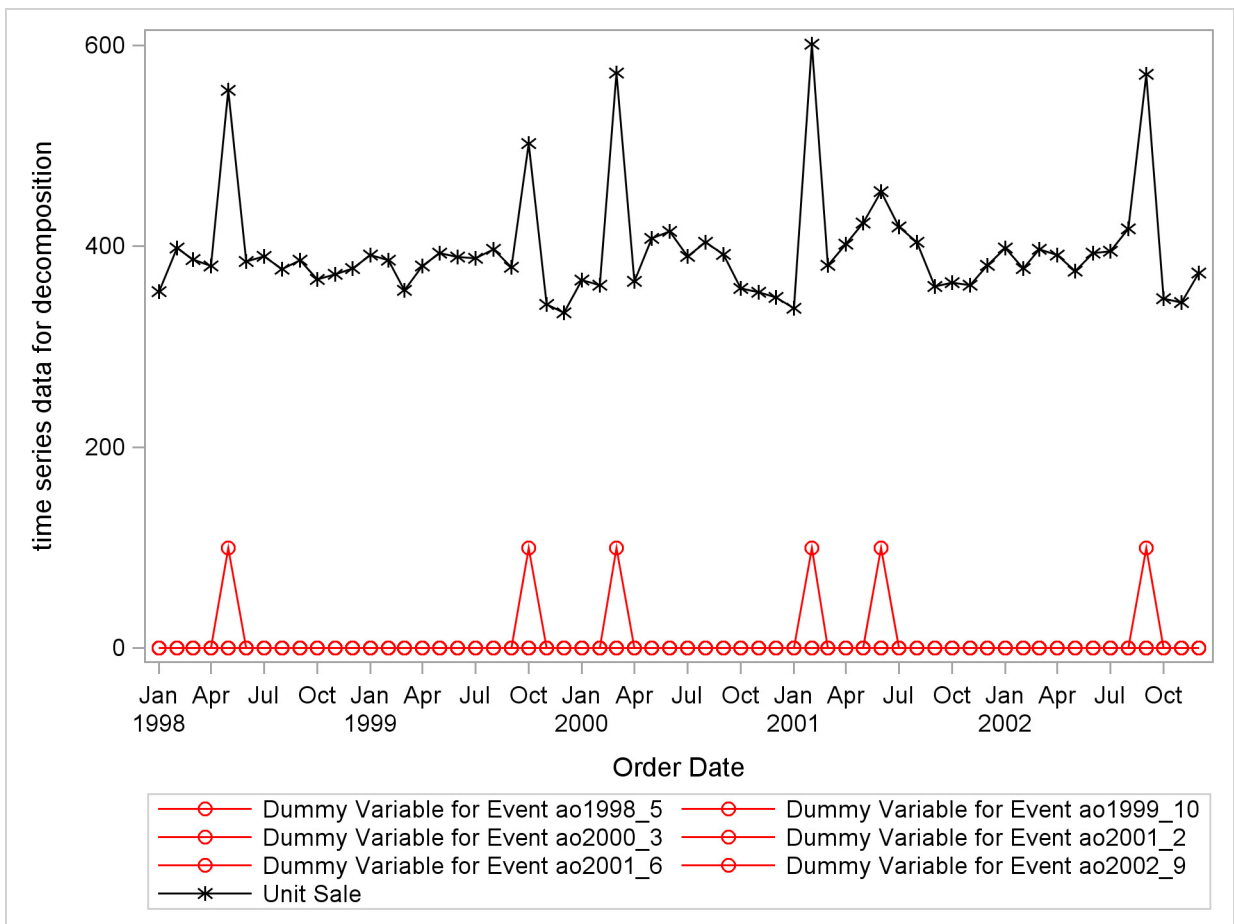
```
proc sgplot data=pid1;
  series x=date y=AO1998_5 / markers markerattrs=(symbol=circle color=red)
  lineattrs=(pattern=1 color=red);
```

```

series x=date y=A01999_10 / markers markerattrs=(symbol=circle color=red)
      lineattrs=(pattern=1 color=red);
series x=date y=A02000_3 / markers markerattrs=(symbol=circle color=red)
      lineattrs=(pattern=1 color=red);
series x=date y=A02001_2 / markers markerattrs=(symbol=circle color=red)
      lineattrs=(pattern=1 color=red);
series x=date y=A02001_6 / markers markerattrs=(symbol=circle color=red)
      lineattrs=(pattern=1 color=red);
series x=date y=A02002_9 / markers markerattrs=(symbol=circle color=red)
      lineattrs=(pattern=1 color=red);
series x=date y=sale / markers markerattrs=(symbol=asterisk color=black)
      lineattrs=(pattern=1 color=black);
yaxis label='time series data for decomposition';
run;

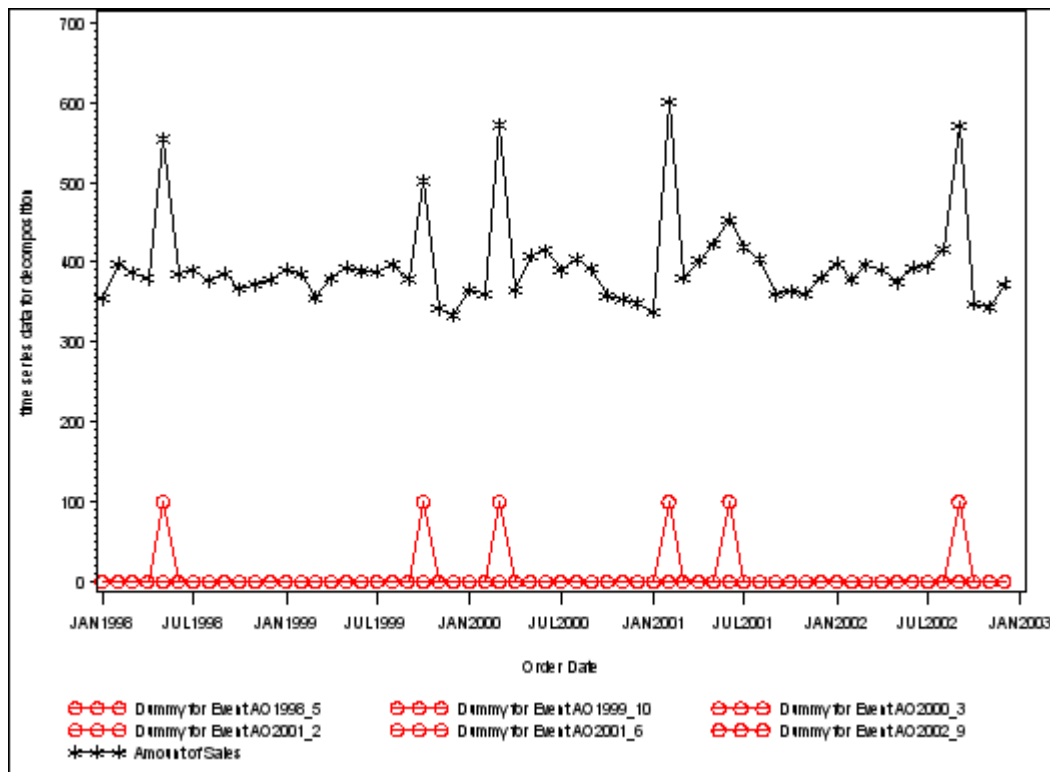
```

Output 6.2.3 Plot of Amount and Dummy





## Output 6.2.3 continued




---

**Example 6.3: 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 have been altered by adding a level shift 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”.

```

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;

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;

proc hpfsselect 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 hpfeengine 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;

```



The output from PROC HPFDIAGNOSE shows that a model was selected that included the level shift occurring at October 1980.

**Output 6.4.1** 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

ARIMA Model Specification											
Variable	Functional Transform	Constant	p	d	q	P	D	Q	Seasonality	Model 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											
Variable	Functional Transform	Constant	p	d	q	P	D	Q	Seasonality	Event	Model 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

Output 6.4.1 *continued*

ARIMA Model Specification After Adjusting for Events											
Variable	Functional Transform	Constant	p	d	q	P	D	Q	Seasonality	Event	Model 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 6.5: Viewing Dummy Variables Using SAS/GRAPH Software

This example illustrates how the HPFEVENTS procedure can be used to create dummies. These dummy variables can then be viewed using SAS/GRAPH software. 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(obs=10);
run;

```

Output 6.5.1 Ramp Dummy Variables

			i	i	i	i	m	m	m	m	m	m	m	m
O	D		n	n	n	n	n	n	n	n	n	n	n	n
b	A	A	f	f	f	f	f	f	f	f	f	f	f	f
s	T	I	g	g	d	d	g	g	d	d	g	g	d	d
	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

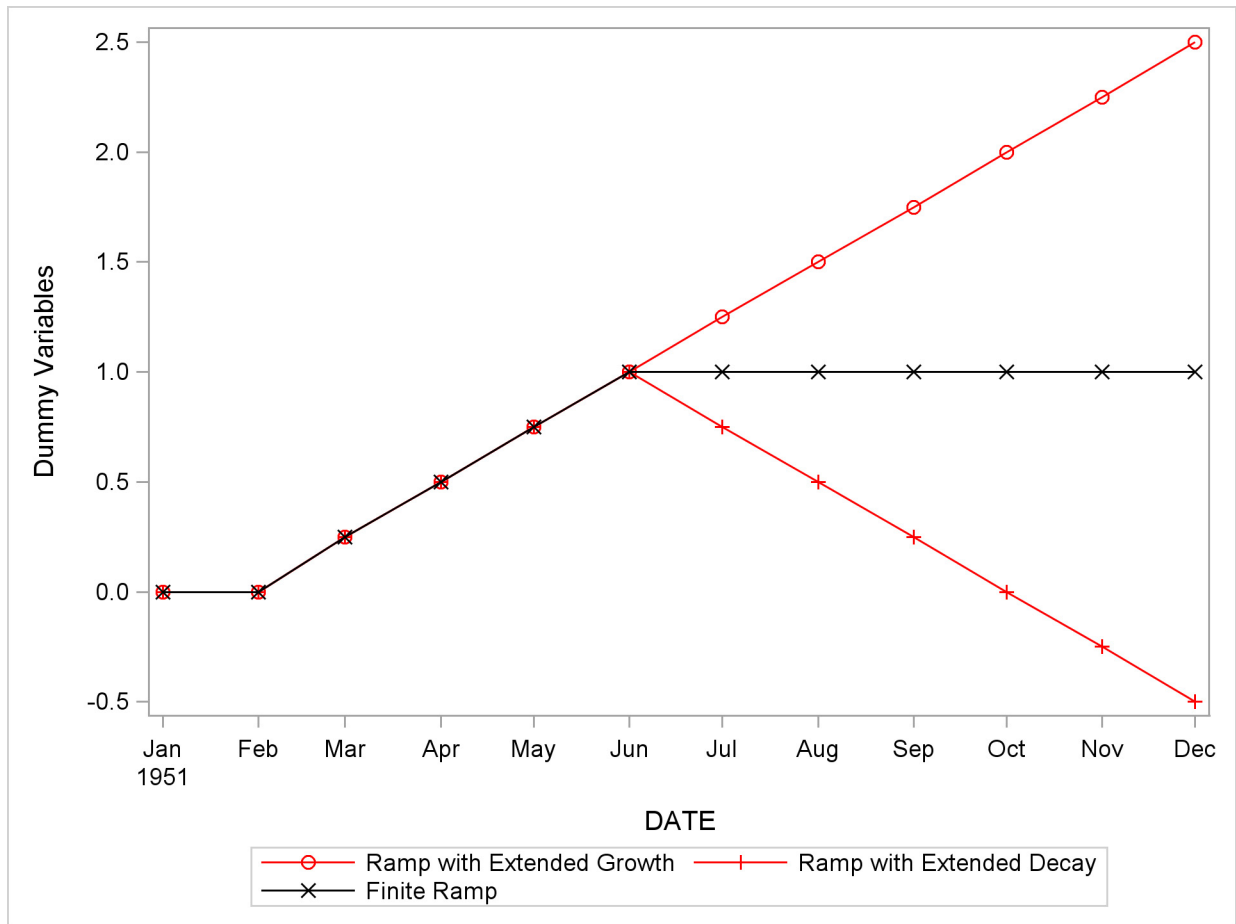
  

					n	n	n	n				
O	n	n	n	n	o	o	o	o				
b	i	i	i	i	n	n	n	n				
s	n	n	n	n	l	l	l	l	m	m	m	m
	f	f	f	f	y	y	y	y	n	n	n	n
	g	g	d	d	g	g	d	d	g	g	d	d
	g	d	g	d	g	d	g	d	g	d	g	d
1	-2.5	-1.5	2.5	3.5	0.0	1.0	0.0	1.0	0.00000	0.00	1.00	1.00000
2	-2.0	-1.0	2.0	3.0	0.0	1.0	0.0	1.0	0.00000	0.00	1.00	1.00000
3	-1.5	-0.5	1.5	2.5	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

```

proc sgplot data=rampdummies;
  series x=date y=minfgg / markers markerattrs=(color=red)
  lineattrs=(pattern=1 color=red) legendlabel='Ramp with Extended Growth';
  series x=date y=minfgd / markers markerattrs=(color=red)
  lineattrs=(pattern=1 color=red) legendlabel='Ramp with Extended Decay';
  series x=date y=monlygg / markers markerattrs=(color=black)
  lineattrs=(pattern=1 color=black) legendlabel='Finite Ramp';
  yaxis label='Dummy Variables';
run;

```

**Output 6.5.2** Plot of Finite and Extended Dummy Variables


---

## References

Montes, M. J. (2001a), *Calculation of the Ecclesiastical Calendar*, <http://www.smart.net/~mmontes/ec-cal.html>.

Montes, M. J. (2001b), *Nature (1876) Algorithm for Calculating the Date of Easter in the Gregorian Calendar*, <http://www.smart.net/~mmontes/nature1876.html>.



# Chapter 7

## The HPFEXMSPEC Procedure

### Contents

---

Overview . . . . .	231
Getting Started . . . . .	232
Syntax . . . . .	232
Functional Summary . . . . .	232
PROC HPFEXMSPEC Statement . . . . .	233
EXM Statement . . . . .	233
Examples . . . . .	235
Example 7.1: Various EXM Model Specifications . . . . .	235

---

---

### 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
Model Repository Options		
specifies the model specification label	PROC HPFEXMSPEC	LABEL=
specifies the model specification name	PROC HPFEXMSPEC	NAME=
specifies the model repository	PROC HPFEXMSPEC	REPOSITORY=
External Model Options		
specifies median forecasts	EXM	MEDIAN

Description	Statement	Option
specifies the method of creating forecast standard errors	EXM	METHOD=
specifies the number of time lags used to compute the autocorrelations	EXM	NLAGPCT=
specifies that the external model parameters are fixed values	EXM	NOEST
specifies the number of parameters used to create the external forecast	EXM	NPARMS=
specifies the prediction standard error for the external model	EXM	SIGMA=
specifies the time series transformation	EXM	TRANSFORM=

---

## 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:

### **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.

### **NLAGPCT=***number*

specifies the number of time lags as a percentage of the number of computed predictions errors. The default is NLAGPCT=0.25.

### **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.

### **NPARMS=***n*

specifies the number of parameters used by the external model to generate the forecasts. The default is NPARMS=0.

### **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.

### **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( <i>n</i> )	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.

---

## Examples

---

### Example 7.1: Various EXM Model Specifications

The following example illustrate typical uses of the EXM statement:

```
proc hpfexmspec repository=mymodels
    name=model1
    label="Default Specification";
    exm;
run;

proc hpfexmspec repository=mymodels
    name=model2
    label="Actual Series Autocorrelation";
    exm method=acf;
run;

proc hpfexmspec repository=mymodels
    name=model3
    label="Prediction Error Autocorrelation";
    exm method=erroracf;
run;

title "Models Added to MYMODELS Repository";
proc catalog catalog=mymodels;
    contents;
run;
```

**Output 7.1.1** Listing of Models in MYMODELS repository

Models Added to MYMODELS Repository					
Contents of Catalog WORK.MYMODELS					
#	Name	Type	Create Date	Modified Date	Description
1	MODEL1	XML	09Jul07:16:31:06	09Jul07:16:31:06	Default Specification
2	MODEL2	XML	09Jul07:16:31:06	09Jul07:16:31:06	Actual Series Autocorrelation
3	MODEL3	XML	09Jul07:16:31:06	09Jul07:16:31:06	Prediction Error Autocorrelation

# Chapter 8

## The HPFIDMSPEC Procedure

### Contents

---

Overview . . . . .	237
Getting Started . . . . .	237
Syntax . . . . .	238
Functional Summary . . . . .	238
PROC HPFIDMSPEC Statement . . . . .	239
IDM Statement . . . . .	240
Smoothing Model Suboptions for IDM Statement Options . . . . .	241
Details . . . . .	245
Smoothing Model Parameter Specification Options . . . . .	245
Smoothing Model Forecast Bounds Options . . . . .	245
Examples . . . . .	245
Example 8.1: Various Kinds of IDM Model Specifications . . . . .	245
Example 8.2: Automatically Choosing the Best Decomposed Demand Model . . . . .	247

---

---

### 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=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
<b>Statements</b>		
specifies the intermittent demand model	<b>IDM</b>	
<b>Model Repository Options</b>		
specifies the model specification label	<b>PROC HPFIDMSPEC</b>	<b>LABEL=</b>
specifies the model specification name	<b>PROC HPFIDMSPEC</b>	<b>NAME=</b>
specifies the model repository	<b>PROC HPFIDMSPEC</b>	<b>REPOSITORY=</b>
<b>Intermittent Demand Model Options</b>		
specifies the model for average demand	<b>IDM</b>	<b>AVERAGE=</b>
specifies the base value	<b>IDM</b>	<b>BASE=</b>
specifies the model for demand intervals	<b>IDM</b>	<b>INTERVAL=</b>
specifies the model for demand sizes	<b>IDM</b>	<b>SIZE=</b>
<b>Exponential Smoothing Model Options</b>		
specifies the model selection criterion	<b>IDM</b>	<b>CRITERION=</b>



Description	Statement	Option
specifies the damping weight parameter initial value	IDM	DAMPPARM=
specifies the damping weight parameter restrictions	IDM	DAMPREST=
specifies the level weight parameter initial value	IDM	LEVELPARG=
specifies the level weight parameter restrictions	IDM	LEVELREST=
specifies median forecasts	IDM	MEDIAN
specifies the time series forecasting model	IDM	METHOD=
specifies that the smoothing model parameters are fixed values	IDM	NOEST
specifies that stable parameter estimates are not required	IDM	NOSTABLE
specifies the time series transformation	IDM	TRANSFORM=
specifies the trend weight parameter initial value	IDM	TRENDPARG=
specifies the trend weight parameter restrictions	IDM	TRENDREST=

## 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 IDM statement can be specified.

The following options can be specified in the IDM statement:

### **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.

### **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.

---

## Smoothing Model Suboptions for IDM Statement Options

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

The following describes the smoothing model options:

**BOUNDS=( number , number )**

Specifies the component forecast bound. See the following smoothing model forecast bounds.

**CRITERION= option**

**SELECT= option**

specifies the model selection criterion (statistic of fit) to be used to select from several candidate models. This option is often used in conjunction with the HOLDOUT= option specified in the FORECAST statement. The CRITERION= option can also be specified as SELECT=. The default is CRITERION=RMSE.

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
MAPES	Mean Absolute Error Percent of Standard Deviation
MDAPES	Median Absolute Error Percent of Standard Deviation
GMAPES	Geometric Mean Absolute Error Percent of Standard Deviation
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
MASE	Mean Absolute Scaled Error
RSQUARE	R-Square
ADJRSQ	Adjusted R-Square
AADJRSQ	Amemiya's Adjusted R-Square
RWRSQ	Random Walk R-Square
AIC	Akaike Information Criterion
AICC	Finite Sample Corrected AIC
SBC	Schwarz Bayesian Information Criterion
APC	Amemiya's Prediction Criterion

**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.

**LEVELPARG=** *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.

**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 CRITERION= 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)

**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.

**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.

**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( <i>n</i> )	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.

**TRENDPARG=** *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.

The following shows the defaults values of the smoothing model options that are used when the options are not specified.

For the demand interval component the defaults are:

```
interval=( transform=auto
           method=bestn
           levelrest=(0.001 0.999)
           trendrest=(0.001 0.999)
           damprest =(0.001 0.999)
           criterion=rmse
           bounds=(1, .)
           )
```

For the demand size component the defaults are:

```
size=(    transform=auto
          method=bestn
          levelrest=(0.001 0.999)
          trendrest=(0.001 0.999)
          damprest =(0.001 0.999)
          criterion=rmse
          )
```

For the average demand component the defaults are:

```
average=( transform=auto
           method=bestn
           levelrest=(0.001 0.999)
           trendrest=(0.001 0.999)
           damprest =(0.001 0.999)
           criterion=rmse
           )
```

---

## Details

---

### 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.

---

## Examples

---

### Example 8.1: Various Kinds of IDM Model Specifications

The following example illustrate typical uses of the IDM statement:

```
proc hpfidmspec repository=mymodels
                name=model1
                label="Default Specification";
    idm;
run;

proc hpfidmspec repository=mymodels
                name=model2
                label="Demand Interval model only specification";
    idm interval=( transform=log );
run;

proc hpfidmspec repository=mymodels
```

```

        name=model3
        label="Demand Size model only specification";
    idm size=( method=linear );
run;

proc hpfidmspec repository=mymodels
    name=model4
    label="Croston's Method";
    idm interval=( method=simple )
        size      =( method=simple );
run;

proc hpfidmspec repository=mymodels
    name=model5
    label="Log Croston's Method";
    idm interval=( method=simple transform=log )
        size      =( method=simple transform=log );
run;

proc hpfidmspec repository=mymodels
    name=model6
    label="Average demand model specification";
    idm average=(method=bestn);
run;

title "Models Added to MYMODELS Repository";
proc catalog catalog=mymodels;
    contents;
run;

```

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 models added to the model repository are shown in [Output 8.1.1](#).

**Output 8.1.1** Listing of Models in MYMODELS repository

Models Added to MYMODELS Repository					
Contents of Catalog WORK.MYMODELS					
#	Name	Type	Create Date	Modified Date	Description
1	MODEL1	XML	09Jul07:16:31:00	09Jul07:16:31:00	Default Specification
2	MODEL2	XML	09Jul07:16:31:00	09Jul07:16:31:00	Demand Interval model only specification
3	MODEL3	XML	09Jul07:16:31:00	09Jul07:16:31:00	Demand Size model only specification
4	MODEL4	XML	09Jul07:16:31:01	09Jul07:16:31:01	Croston's Method
5	MODEL5	XML	09Jul07:16:31:01	09Jul07:16:31:01	Log Croston's Method
6	MODEL6	XML	09Jul07:16:31:01	09Jul07:16:31:01	Average demand model specification
7	MYCROSTO	XML	09Jul07:16:31:00	09Jul07:16:31:00	Croston Method
	N				



## Example 8.2: Automatically Choosing the Best Decomposed Demand Model

This example illustrates how to automatically choose the decomposed demand model using mean absolute percent error (MAPE) as the model selection criterion.

```
proc hpfidmspec repository=mymodels
    name=auto1
    label="Automatically Selected Best IDM Model";
    idm interval=( method=simple transform=auto criterion=mape )
    size      =( method=simple transform=auto criterion=mape );
run;
```

The preceding model specification causes PROC HPFENGINE to fit 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 illustrates how to automatically choose the average demand model using MAPE as the model selection criterion.

```
proc hpfidmspec repository=mymodels
    name=auto2
    label="Automatically Selected Best IDM Model";
    idm average=( method=simple transform=auto criterion=mape );
run;
```

The preceding specification causes PROC HPFENGINE to fit 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:

```
proc hpfidmspec repository=mymodels
    name=auto3
    label="Automatically Selected Best IDM Model";
    idm interval=( method=simple transform=auto criterion=mape )
    size      =( method=simple transform=auto criterion=mape )
    average   =( method=simple transform=auto criterion=mape );
run;
```

The preceding model specification causes PROC HPFENGINE to automatically select 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.



# Chapter 9

## The HPFRECONCILE Procedure

### Contents

---

Overview: HPFRECONCILE Procedure . . . . .	249
Getting Started: HPFRECONCILE Procedure . . . . .	250
Syntax: HPFRECONCILE Procedure . . . . .	251
Functional Summary . . . . .	252
PROC HPFRECONCILE Statement . . . . .	254
AGGBY Statement . . . . .	257
AGGDATA Statement . . . . .	257
BY Statement . . . . .	258
DISAGGDATA Statement . . . . .	258
ID Statement . . . . .	259
Details: HPFRECONCILE Procedure . . . . .	260
Notation . . . . .	260
Top-Down Reconciliation . . . . .	261
Bottom-Up Reconciliation . . . . .	265
Data Set Input/Output . . . . .	266
Examples: HPFRECONCILE Procedure . . . . .	273
Example 9.1: Reconciling a Hierarchical Tree . . . . .	273
Example 9.2: Aggregating Forecasts . . . . .	276
Example 9.3: Disaggregating Forecasts . . . . .	277
Example 9.4: Imposing Constraints . . . . .	279

---

---

### Overview: HPFRECONCILE Procedure

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 total sales of a product by a retail company are 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. Imposing such constraints during the forecasting process can be difficult or impossible. Therefore, the series are often forecast independently at different levels. The resulting forecasts, however, do not abide by the constraints binding the original series. The after-the-fact process through which such constraints are enforced is called *reconciliation*.

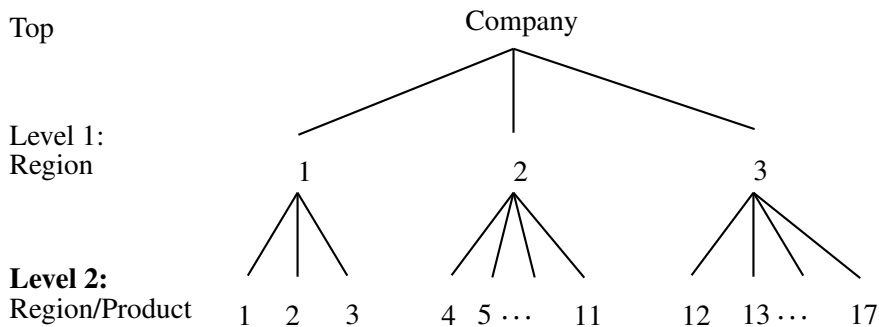
The HPFRECONCILE procedure reconciles forecasts of time series data at two different levels of a hierarchy. Optionally, the HPFRECONCILE procedure can disaggregate forecasts from upper-level forecasts or aggregate forecasts from lower-level forecasts. The procedure enables the user to specify the direction and the method of reconciliation, equality constraints, and bounds on the reconciled values.

## Getting Started: HPFRECONCILE Procedure

This section outlines the use of the HPFRECONCILE procedure.

Consider the following hierarchical structure of the SASHELP.PRICEDATA data set.

**Figure 9.1** 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=lv12est
    modelrepository=work.mymodels
    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=lv12est
    out=_null_
    outest=lv12fest
    modelrepository=mymodels
  
```

```

        outfor=lv12for;
    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=lv1lest
    modelrepository=work.mymodels
    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=lv1lest
    out=_null_
    outest=lv1lfest
    modelrepository=mymodels
    outfor=lv1lfor;
    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=lv12for
    aggdata=lv1lfor
    direction=BU
    outfor=lv1lrecfor;
    id date interval=month;
    by region product;
run;

```

---

## Syntax: HPFRECONCILE Procedure

The HPFRECONCILE procedure is controlled by the following statements:

```

PROC HPFRECONCILE < options > ;
  AGGBY variables < / option > ;
  AGGDATA < options > ;
  BY variables < / option > ;
  DISAGGDATA < options > ;
  ID variable INTERVAL=interval < / options > ;

```

---

## Functional Summary

The statements and options used with the HPFRECONCILE procedure are summarized in the following table.

Description	Statement	Option
<b>Statements</b>		
specify the AGGBY variables	<b>AGGBY</b>	
names variables in the AGGDATA= data set	<b>AGGDATA</b>	
specify the BY variables	<b>BY</b>	
names variables in the DISAGGDATA= data set	<b>DISAGGDATA</b>	
specify the time ID variable	<b>ID</b>	
<b>Data Set Options</b>		
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 containing the reconciled forecasts	HPFRECONCILE	OUTFOR=
specify the output data set containing information regarding the infeasible problems	HPFRECONCILE	OUTINFEASIBLE=
specify the output data set containing summary information	HPFRECONCILE	OUTPROCINFO=
specify the data set containing constraints on the reconciled forecasts	HPFRECONCILE	CONSTRAINT=
specify that constraints be forced on the <i>predict</i> variable in the OUTFOR= data set when conflicting with the aggregation constraint	HPFRECONCILE	FORCECONSTRAINT
specify that the OUTFOR= data sets contain the RECDIFF variable	HPFRECONCILE	RECDIFF
names the variable containing the actual values in the DISAGGDATA= data set	DISAGGDATA	ACTUAL=
names the variable containing the actual values in the AGGDATA= data set	AGGDATA	ACTUAL=

Description	Statement	Option
names the variable containing the predicted values in the DISAGGDATA= data set	DISAGGDATA	PREDICT=
names the variable containing the predicted values in the AGGDATA= data set	AGGDATA	PREDICT=
names the variable containing the lower confidence limit in the DISAGGDATA= data set	DISAGGDATA	LOWER=
names the variable containing the lower confidence limit in the AGGDATA= data set	AGGDATA	LOWER=
names the variable containing the upper confidence limit in the DISAGGDATA= data set	DISAGGDATA	UPPER=
names the variable containing the upper confidence limit in the AGGDATA= data set	AGGDATA	UPPER=
names the variable containing the prediction error in the DISAGGDATA= data set	DISAGGDATA	ERROR=
names the variable containing the prediction error in the AGGDATA= data set	AGGDATA	ERROR=
names the variable containing the standard error in the DISAGGDATA= data set	DISAGGDATA	STD=
names the variable containing the standard error in the AGGDATA= data set	AGGDATA	STD=
<b>Error Message Options</b>		
specify the resolution of error and warning messages	HPFRECONCILE	ERRORTRACE=
<b>Analysis Options</b>		
specify the aggregation method	HPFRECONCILE	AGGREGATE=
specify the confidence level	HPFRECONCILE	ALPHA=
specify method for confidence limits	HPFRECONCILE	CLMETHOD=
specify the reconciliation direction	HPFRECONCILE	DIRECTION=
specify the starting time ID value	ID	START=
specify the ending time ID value	ID	END=
specify the frequency	ID	INTERVAL=
specify the alignment of time ID values	ID	ALIGN=
specify the disaggregation function	HPFRECONCILE	DISAGGREGATION=
specify that only the prediction be reconciled	HPFRECONCILE	PREDICTONLY
specify sign constraint on the reconcile series	HPFRECONCILE	SIGN=
specify method of computing standard errors	HPFRECONCILE	STDMETHOD=
specify bounds for the standard error	HPFRECONCILE	STDDIFBD=
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

**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.

See “[AGGDATA= Data Set](#)” on page 266 for more details on the AGGDATA= data set.

**CONSTRAINT=** *SAS-data-set*

specifies the name of the SAS data set containing the constraints for the reconciled series. See “[CONSTRAINT= Data Set](#)” on page 268 for more details.

**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.

See “[DISAGGDATA= Data Set](#)” on page 267 for more details on the DISAGGDATA= data set.

### Options Related to the Output Data Sets

**FORCECONSTRAINT**

specifies whether the user-specified constraints should be forced on the PREDICT variable in the OUTFOR= data set when the problem is infeasible because the constraints are incompatible with the aggregation constraint. The default is to leave the input unmodified.

**OUTFOR=** *SAS-data-set*

specifies the name of the output SAS data set containing the reconciled values.

**OUTINFEASIBLE=** *SAS-data-set*

specifies the name of the SAS data set containing a summary of the nodes for which rec-



conciliation failed because of a conflict between the constraints imposed by the user and the aggregation constraint.

**OUTPROCINFO=** *SAS-data-set*

names the output data set to contain the summary information of the processing done by PROC HPFRECONCILE. It is particularly useful for easy programmatic assessment of the status of the procedure's execution via a data set instead of looking at or parsing the SAS log.

**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:

DATASET	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=DATASET.

## 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 0.0001 and 0.9999. When you specify ALPHA= $\alpha$ , the upper and lower confidence limits will have a  $1 - \alpha$  confidence level. The default is ALPHA=.05, which produces 95% confidence intervals.

**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: HPFRECONCILE Procedure](#)” on page 260 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: HPFRECONCILE Procedure](#)” on page 260 for more information about the reconciliation directions available in PROC HPFRECONCILE.

**DISAGGREGATION= DIFFERENCE | PROPORTIONS**

specifies the type of loss function for top-down reconciliation.

DISAGGREGATION=PROPORTIONS is available only when all the forecasts at a given ID value share the same sign. See “[Details: HPFRECONCILE Procedure](#)” on page 260 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:

MIXED if the output series can have any sign. This is the default.

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=UNCHANGED. See “[Details: HPFRECONCILE Procedure](#)” on page 260 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.

**WEIGHTED**

specifies that the loss function for top-down reconciliation be weighted by the inverse of the variance of the input forecasts.

---

## 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.

---

## AGGDATA Statement

**AGGDATA < *options* > ;**

The AGGDATA statement enables the user to specify custom names for forecasting variables in the AGGDATA= data set. The default names are ACTUAL, PREDICT, LOWER, UPPER, ERROR, and STD.

The following options may be specified on the AGGDATA statement.

**ACTUAL= *variable-name***

specifies the name of the variable in the AGGDATA= data set that contains the actual values.

**PREDICT= *variable-name***

specifies the name of the variable in the AGGDATA= data set that contains the predicted values.

**LOWER= *variable-name***

specifies the name of the variable in the AGGDATA= data set that contains the lower confidence limit values.

**UPPER= *variable-name***

specifies the name of the variable in the AGGDATA= data set that contains the upper confidence limit values.

**ERROR=** *variable-name*

specifies the name of the variable in the AGGDATA= data set that contains the error values.

**STD=** *variable-name*

specifies the name of the variable in the AGGDATA= data set that contains the standard error values.

---

## 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 strict 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. Since the AGGBY variables form a proper subset of the BY variables, their number must be less than the number of BY 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.

---

## DISAGGDATA Statement

**DISAGGDATA** < *options* > ;

The DISAGGDATA statement enables you to specify names for forecasting variables in the DISAGGDATA= data set. The default names are ACTUAL, PREDICT, LOWER, UPPER, ERROR, and STD.

The following options can be specified on the DISAGGDATA statement.

**ACTUAL=** *variable-name*

specifies the name of the variable in the DISAGGDATA= data set that contains the actual values.

**PREDICT=** *variable-name*

specifies the name of the variable in the DISAGGDATA= data set that contains the predicted values.

**LOWER=** *variable-name*

specifies the name of the variable in the DISAGGDATA= data set that contains the lower confidence limit values.

**UPPER=** *variable-name*

specifies the name of the variable in the DISAGGDATA= data set that contains the upper confidence limit values.

**ERROR=** *variable-name*

specifies the name of the variable in the DISAGGDATA= data set that contains the error values.

**STD=** *variable-name*

specifies the name of the variable in the DISAGGDATA= data set that contains the standard error values.

---

## 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, and the ID variable must be present and must have the same frequency in both the DISAGGDATA= data set and the AGGDATA= data set. If an ID statement is not specified, then a number derived from the observation number, with respect to the BY group, is used as the time ID. The number is derived so as to align the last observations of all BY groups.

The following options can be used with the ID statement.

**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.

**IRREGULAR**

specifies whether to allow for irregularities in the ID variable frequency. By default, 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.

**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 forecasts that are outside the historical period. For example, START=“&sysdate”D uses the automatic macro variable SYSDATE to start the reconciliation at the current date.

**ALIGN=** *option*

controls the alignment of SAS dates used to identify output observations. Internal processing uses aligned versions of the values of START= and END= options (if specified) and values of ID variable in input observations. The ALIGN= option accepts the following values: BEGIN, MIDDLE, END. BEGIN is the default.

---

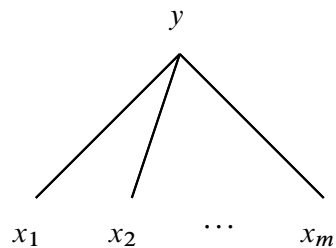
## Details: HPFRECONCILE Procedure

---

### Notation

Assume a two-level hierarchical structure as depicted in Figure 9.2.

**Figure 9.2** Hierarchical Structure



Let  $y_t$  be the values of the parent series at time  $t$ , and let  $\mathbf{x}_t = [x_{1,t}, x_{2,t}, \dots, x_{m,t}]'$  be the vector child series at time  $t$ ,  $t = 1, \dots, T$ . As usual, indicate by  $\hat{y}_t$  and  $\hat{\mathbf{x}}_t$  the pre-reconciliation forecasts of  $y_t$  and  $\mathbf{x}_t$ , respectively, and denote by  $\hat{\boldsymbol{\sigma}}_t = [\hat{\sigma}_{1,t}, \hat{\sigma}_{2,t}, \dots, \hat{\sigma}_{m,t}]'$  the vector of prediction standard error for  $\hat{\mathbf{x}}_t$ . Denote by  $\hat{\boldsymbol{\Sigma}}$  the diagonal matrix whose main diagonal is  $\hat{\boldsymbol{\sigma}}_t^2$ . Let the superscript tilde indicate the reconciled values, so that  $\tilde{y}_t$  and  $\tilde{\mathbf{x}}_t$  indicate the reconciled values of  $\hat{y}_t$  and  $\hat{\mathbf{x}}_t$ , respectively. The number of child 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. 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. The aggregation constraint is  $y_t = \sum_{i=1}^m x_{i,t}$ , when you specify the AGGREGATE=TOTAL option of the PROC HPFRECONCILE statement. If, instead, you specify the AGGREGATE=AVERAGE option, the constraint is  $y_t = \frac{1}{m} \sum_{i=1}^m x_{i,t}$ .

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. 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

$$\tilde{x}_{i,t} = e_{i,t} \quad i \in E_t$$

3. the lower bounds

$$\tilde{x}_{i,t} \geq l_{i,t} \quad i \in L_t$$

4. the upper bounds

$$\tilde{x}_{i,t} \leq u_{i,t} \quad i \in U_t$$

where  $E_t$ ,  $L_t$ , and  $U_t$  are subsets of  $\{1, 2, \dots, m\}$ .

Equality constraints on the reconciled predicted values can be imposed with the EQUALITY variable of the CONSTRAINT= data set, or by using the WEIGHTED option with zero standard errors in the DISAGGDATA= data set. Bounds can be imposed either with the SIGN= option or with the LOWER and UPPER variables of the CONSTRAINT= data set.

PROC HPFRECONCILE employs an iterative interior point algorithm to solve the constrained quadratic optimization problem.

## Choice of Loss Function

The loss function takes the following functional forms:

- When DISAGGREGATION=DIFFERENCE, the loss function is

$$L(\tilde{\mathbf{x}}_t; \hat{\mathbf{x}}_t) = (\tilde{\mathbf{x}}_t - \hat{\mathbf{x}}_t)' W^{-1} (\tilde{\mathbf{x}}_t - \hat{\mathbf{x}}_t)$$

- When DISAGGREGATION=PROPORTIONS, 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  $W$  is a positive semidefinite matrix of weights independent of  $\tilde{\mathbf{x}}_t$ , and  $\hat{X}^{-\frac{1}{2}}$  is a diagonal matrix with the square root of  $\hat{\mathbf{x}}_t$  on the main diagonal.

If the WEIGHTED option is not specified,  $W$  is the identity matrix  $I$ . If the WEIGHTED option is specified,  $W = \hat{\Sigma}$ , the diagonal matrix with the estimated variances  $\hat{\sigma}_{i,t}^2$  of  $\hat{x}_{i,t}$  on the main diagonal. If an observation has zero prediction standard error,  $\hat{\sigma}_{j,t}^2 = 0$ , it is equivalent to imposing a locked equality constraint equal to the original forecast,  $\tilde{x}_{j,t} = \hat{x}_{j,t}$ .

When DISAGGREGATION=DIFFERENCE, the loss function is defined for any value of  $\hat{\mathbf{x}}_t$ .

When DISAGGREGATION=PROPORTIONS, the loss function is defined only when all  $\hat{x}_{i,t}$  are strictly positive. The solutions can be extended to the cases where they are all nonnegative by letting  $\tilde{x}_{i,t} := 0$  when  $\hat{x}_{i,t} = 0$ , if there is at least one  $\hat{x}_{j,t}$  that is greater than zero. The case where  $\sum_{j=1}^m \hat{x}_{j,t} = 0$  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. The solution can be further extended to the case when all  $\hat{x}_{i,t}$  are nonpositive, by defining  $\hat{z}_{i,t} := -\hat{x}_{i,t}$ ,  $i = 1, \dots, m$ , and  $\tilde{x}_{i,t} := -\tilde{z}_{i,t}$ ,  $i = 1, \dots, m$ .

If DISAGGREGATION=PROPORTIONS, 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 that 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.

## Unconstrained Solutions

When the only constraint is the top-down constraint and  $W = I$ , 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:

$$\tilde{x}_{i,t} = \hat{x}_{i,t} + \frac{\hat{r}_t}{m}$$



where  $\hat{r}_t$  is the forecasting aggregate error—that is,

$$\hat{r}_t := \hat{y}_t - \sum_{i=1}^m \hat{x}_{i,t}$$

when AGGREGATE=TOTAL and

$$\hat{r}_t := m\hat{y}_t - \sum_{i=1}^m \hat{x}_{i,t}$$

when AGGREGATE=AVERAGE.

Thus, when DISAGGREGATION=DIFFERENCE, the reconciled forecast  $\tilde{x}_{i,t}$  is found by equally splitting the aggregation error  $\hat{r}_t$  among the lower-level 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:

$$\tilde{x}_{i,t} = \frac{\hat{x}_{i,t}}{\sum_{j=1}^m \hat{x}_{j,t}} \hat{y}_t$$

when AGGREGATE=TOTAL, and

$$\tilde{x}_{i,t} = \frac{\hat{x}_{i,t}}{\sum_{j=1}^m \hat{x}_{j,t}} m\hat{y}_t$$

when AGGREGATE=AVERAGE.

Thus, the reconciled forecast  $\tilde{x}_{i,t}$  is found by disaggregating the upper-level forecasts according to the proportion that  $\hat{x}_{i,t}$  represents in the total sum of the lower-level forecasts.

## Missing Values

Missing values of type “F” in the input are interpreted as failed forecasts. If a “F” missing value is detected in any of the variables that are needed to compute the reconciled prediction, then the reconciled prediction too will take the value “F”.

When some of the predicted values  $\hat{x}_{i,t}$  are missing, with any type of missing values different from “F”, 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. If the actual value too is missing, the series is excluded from the reconciliation process.

When you use the WEIGHTED option and the standard error is missing, the weight is assumed to be the average of the non-missing variances. If all standard errors are missing, the weights are assumed to be all equal to one, which is equivalent to not using the WEIGHTED option.

## 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,

$$\tilde{\sigma}_{i,t} = |\hat{p}_{i,t}|\hat{\sigma}_t$$

where  $\hat{p}_{i,t} = \frac{\tilde{x}_{i,t}}{\hat{y}_t}$ , 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.

Care should be taken in interpreting standard errors when constraints are imposed on the reconciled forecasts. The presence of constraints renders the meaning of the reconciled standard errors ambiguous.

## 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, `CLMETHOD=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.

Care should be used in interpreting confidence limits when constraints are imposed on the reconciled forecasts. The presence of constraints renders the meaning of reconciled confidence limits ambiguous.

## 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

$$\tilde{y}_t = \sum_{i=1}^m \hat{x}_{i,t} \quad t = 1, 2, \dots$$

When AGGREGATE=AVERAGE, this is done by setting

$$\tilde{y}_t = \frac{1}{m} \sum_{i=1}^m \hat{x}_{i,t} \quad t = 1, 2, \dots$$

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 OUTPUT= data set.

## Missing Predicted Values

Missing values of type “.F” in the input are interpreted as failed forecasts. If a “.F” missing value is detected in any of the variables that are needed to compute the reconciled prediction, then the reconciled prediction too will take the value “.F”.

When some of the predicted values  $\hat{x}_{i,t}$  are missing, with missing value different from “.F”, 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 for a given time ID  $t$ , then the reconciliation process is considered failed for this  $t$ , even though the actual values  $x_{i,t}$  are not 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

$$\tilde{\sigma}_t = \sqrt{\sum_{i=1}^m \hat{\sigma}_{i,t}^2}$$

when AGGREGATE=TOTAL, and

$$\tilde{\sigma}_t = \frac{1}{m} \sqrt{\sum_{i=1}^m \hat{\sigma}_{i,t}^2}$$

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, CLMETHOD=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

### AGGDATA= Data Set

The AGGDATA= data set contains a proper 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 4, “[The HPFENGINE Procedure](#),” for more details.

The BY variables contained in the AGGDATA= data set are called AGGBY variables. The AGGDATA= data set must be either sorted by the AGGBY variables and by the ID variable when the latter is specified, or indexed on the AGGBY variables. Even when the data set is indexed, if the ID variable is specified, its values must be sorted in ascending order within each AGGBY group.

You can specify custom names for the variables in the AGGDATA= data set by using the [AGGDATA statement](#).

## 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:

<u>_NAME_</u>	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 4, “[The HPFENGINE Procedure](#),” for more details.

The DISAGGDATA= data set must be either sorted by the BY variables and by the ID variable when the latter is specified, or indexed on the BY variables. If the variable \_NAME\_ is present and has multiple values, then the index must be a composite index on BY variables and \_NAME\_, in that order. If \_NAME\_ is present and has only one value, then the index can contain only BY variables. Even when the data set is indexed, if the ID variable is specified, its values must be sorted in ascending order within each BY, or BY and \_NAME\_ group, as applicable. Indexing the DISAGGDATA= data set on the BY variables when it is already sorted by the BY variables will lead to less efficient and less scalable operation if the available memory is not sufficient to hold the disaggregated data for the AGGBY group being processed. The amount of memory required depends on, among other things, the length of the series, the number of BY groups for each AGGBY group, and the number and format of the BY variables. For example, if there are four BY variables, each 16 characters long, 10,000 BY groups within each AGGBY group, and each series has length 100, then the minimum required memory for efficient processing is approximately 100 MB. If the memory is not sufficient, sorting the DISAGGDATA= data set, not indexing, will be more efficient.

You can specify custom names for the variables in the DISAGGDATA= data set by using the DISAGGDATA 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.
	When EQUALITY is nonmissing and the UNLOCK flag is missing, the equality is treated as locked.
LOWERBD	Lower bounds for the reconciled forecasts
UPPERBD	Upper bounds for the reconciled forecasts

Locked equality constraints are treated as constraints, and, therefore, their value is honored. Unlocked equalities are instead treated as regular forecasts and, in general, are changed by the reconciliation process.

A constraint is said to be *active* when the reconciled prediction lies on the constraint. By definition, locked equalities are always active constraints.

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 `OUTFOR=` 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:

<code>_NAME_</code>	Variable name
<code>ACTUAL</code>	Actual values
<code>PREDICT</code>	Predicted values
<code>LOWER</code>	Lower confidence limits
<code>UPPER</code>	Upper confidence limits
<code>ERROR</code>	Prediction errors
<code>STD</code>	Prediction standard errors
<code>_RECONSTATUS_</code>	Reconciliation status

The `OUTFOR=` data set will always be sorted by the `BY` variables, and the `_NAME_` variable and time ID variable when these variables are present, even if input data sets are indexed and not sorted.

If the `ID` statement is specified, then the values of the `ID` variable in `OUTFOR=` data set are aligned based on the `ALIGN=` and `INTERVAL=` options specified on the `ID` statement. If `ALIGN=` option is not specified, then the values are aligned to the beginning of the interval.

If the `RECDIFF` option of the `HPFRECONCILE` statement has been specified, the `OUTFOR=` data sets will also contain the following variable:

<code>RECDIFF</code>	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 `ERROR-TRACE=` 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:

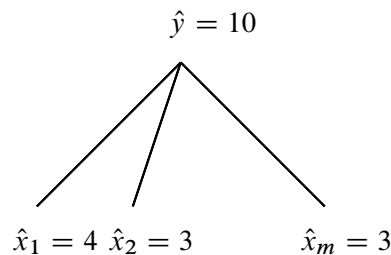
0	Success
400	A unlocked equality constraint has been imposed.
500	A locked equality constraint has been imposed.
600	A lower bound is active.
700	An upper bound is active.
1000	ID value out of the range with respect to the <code>START=</code> and <code>END=</code> interval.
2000	Insufficient data to 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.
- 12000 The number of missing values in the STD variable in the DISAGGDATA= data set is different from the number of missing values in the union of the PREDICT and ACTUAL variables.
- 13000 The solution may be suboptimal. This means that the optimizer did not find an optimal solution, but the solution provided satisfies all constraints.
- 14000 A failed forecast “.F” has been detected in a relevant input variable.

### The FORCECONSTRAINT Option

The FORCECONSTRAINT option applies when there are conflicts between the aggregation constraint and one or more constraints that you specify using either the CONSTRAINT= data set, the SIGN= option, or the WEIGHTED option with zero weights. By default, when reconciliation is impossible, PROC HPFRECONCILE copies the input to the OUTFOR= data set without modification. However, if the reconciliation is infeasible because of a conflict between the constraints you specified and the aggregation constraint, you can ask PROC HPFRECONCILE to impose your constraints on the output even though that results in a violation of the aggregation constraint. For example, assume the input is described by the diagram in Figure 9.3.

**Figure 9.3** FORCECONSTRAINT Option



And assume you want to impose the following constraints on the reconciled forecasts:  $\tilde{x}_1 = 7$ ,  $\tilde{x}_2 \geq 5$ ,  $\tilde{x}_3 \geq 0$ . The constraints are clearly in conflict the aggregation constraint  $\sum \tilde{x}_i = \hat{y}$ ; therefore, PROC HPFRECONCILE will consider the problem infeasible. If you do not specify the FORCECONSTRAINT option, the predicted values in the OUTFOR= data set will equal the input predicted values (that is,  $\tilde{x}_1 = 4$ ,  $\tilde{x}_2 = 3$ ,  $\tilde{x}_3 = 3$ ) and the \_RECONSTATUS\_ variable will take the value 6000. If you specify the FORCECONSTRAINT option, the OUTFOR= data set will contain the values  $\tilde{x}_1 = 7$ ,  $\tilde{x}_2 = 5$ ,  $\tilde{x}_3 = 0$ .



## OUTINFEASIBLE= Data Set

The OUTINFEASIBLE= data set contains summary information about the nodes in the hierarchy for which reconciliation is infeasible because the aggregation constraint is incompatible with the constraints supplied by the user.

The OUTINFEASIBLE= data set is always produced at the level of the AGGDATA= data set.

The OUTINFEASIBLE= data set contains the AGGBY variables present in the AGGDATA= data set, the time ID variable, when it is specified, and the following variables:

<code>_NAME_</code>	Variable name
<code>ISRECONCILED</code>	This variable takes value 1 when the node is reconciled, and value 0 when it is not.
<code>FINALPREDICT</code>	The predicted value for the parent node
<code>AGGCHILDPREDICT</code>	The aggregated prediction of the children nodes
<code>LOWERBD</code>	The lower bound implied by the constraints on FINALPREDICT
<code>UPPERBD</code>	The upper bound implied by the constraints on FINALPREDICT

If the ID statement is specified, then the values of the ID variable in OUTINFEASIBLE= data set are aligned based on the ALIGN= and INTERVAL= options specified on the ID statement. If ALIGN= option is not specified, then the values are aligned to the beginning of the interval.

## OUTNODESUM= Data Set

The OUTNODESUM= data set contains the BY variables in the AGGDATA= data set (or in the AGGBY statement, if the AGGDATA= data set is not specified), the time ID variable in the ID statement when this statement is specified, and the following variables:

<code>_NAME_</code>	Variable name
<code>NONMISSCHLD</code>	Number of nonmissing children of the current AGGBY group

## OUTPROCINFO= Data Set

The OUTPROCINFO= data set contains the following variables:

<code>_SOURCE_</code>	The source procedure that produces this data set
<code>_STAGE_</code>	Stage of the procedure execution for which the summary variable is reported
<code>_NAME_</code>	Name of the summary variable
<code>_LABEL_</code>	Description of the summary variable
<code>_VALUE_</code>	Value of the summary variable

For PROC HPFRECONCILE, the value of the `_SOURCE_` variable is **HPFRECONCILE** and the value of the `_STAGE_` variable is **ALL** for all observations. It contains observations corresponding to each of the following values of the `_NAME_`.

<code>NOBS_RECON</code>	Total number of observations subject to reconciliation
<code>NOBS_SUCCESS</code>	Number of observations with successful reconciliation
<code>NOBS_PREDICTFAIL</code>	Number of observations for which reconciliation failed for PREDICT. This number does not include failures to reconcile due to an infeasible problem or a failed (“F”) forecast.
<code>NOBS_PROBLEMSTATUS</code>	Number of observations for which some problem was encountered. This is the number of observations in OUTFOR= data set that have a <code>_RECONSTATUS_</code> value greater or equal to 1000.
<code>NOBS_INFEASIBLE</code>	Number of observations for which reconciliation is infeasible due to incompatible constraints
<code>NOBS_SUBOPTIMAL</code>	Number of observations for which the optimizer did not find an optimal solution
<code>NOBS_LOCKEQ</code>	Number of observations subject to a locked equality constraint
<code>NOBS_USER_LOCKEQ</code>	Number of observations for which a locked equality was specified in the CONSTRAINT= data set
<code>NOBS_LOWERBD</code>	Number of observations for which a lower bound was imposed
<code>NOBS_USER_LOWERBD</code>	Number of observations for which a lower bound was specified in the CONSTRAINT= data set
<code>NOBS_UPPERBD</code>	Number of observations for which an upper bound was imposed
<code>NOBS_USER_UPPERBD</code>	Number of observations for which an upper bound was specified in the CONSTRAINT= data set
<code>NOBS_ACTIVE_LOWERBD</code>	Number of observations for which a lower bound is active
<code>NOBS_ACTIVE_UPPERBD</code>	Number of observations for which an upper bound is active
<code>NOBS_FAILED_FORECAST</code>	Number of observations for which a failed forecast (“F”) was written
<code>NPROB_TOTAL</code>	Total number of possible problems. One reconciliation problem is possible for each distinct value of time ID variable appearing in AGGDATA= or DISAGGDATA= data sets.
<code>NPROB_CHANGED_LOSS</code>	Number of reconciliation problems for which DISAGGREGATION= option was changed internally because the supplied or default option was not feasible

NPROB_CHANGED_CLMETHOD	Number of reconciliation problems for which CLMETHOD= option was changed internally because the supplied or default option was not feasible
NPROB_CHANGED_STDMETHOD	Number of reconciliation problems for which STDMETHOD= option was changed internally because the supplied or default option was not feasible
NPROB_RECON	Total number of problems subject to reconciliation
NPROB_INFEASIBLE	Number of infeasible reconciliation problems due to incompatible constraints
NPROB_SUBOPTIMAL	Number of reconciliation problems for which the optimizer did not find an optimal solution
NPROB_OPTIMIZER	Number of reconciliation problems solved using the optimizer
ID_MIN	Minimum value of time ID for which reconciliation was attempted
ID_MAX	Maximum value of time ID for which reconciliation was attempted
NAGGBY	Total number of AGGBY groups processed
AVGDBY_PERABY	Average number of DISBY groups per AGGBY group
AVGACTIVEDBY_PERID	Average number of active DISBY groups per time ID for which reconciliation was attempted
NCONS_READ	Total number of constraints read in the CONSTRAINT= data set
NCONS_IN_VALID_ID_RANGE	Total number of constraints in the CONSTRAINT= data set with time ID values in the [START=,END=] range
NCONS_USED	Number of constraints used
CONS_ISANYUNMATCHED	If any of the constraints in the CONSTRAINT= data set is left unmatched and unprocessed, then _VALUE_ for this observation is set to 1; otherwise, it is set to 0.
RC	Return code of PROC HPFRECONCILE

---

## Examples: HPFRECONCILE Procedure

---

### Example 9.1: 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 9.1](#). 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: HPFRECONCILE Procedure](#)” on page 250 section, 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=lv10est
                modelrepository=work.mymodels
                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=lv10est
              out=_null_
              outest=lv10fest
              modelrepository=mymodels
              outfor=lv10for;
    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 lv1recfor 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=lv11for
                 aggdata=lv10for
                 direction=TD
                 outfor=lv11recfor;
    id date interval=month;
    by region;
run;

/* Reconcile forecasts top down from region to region/product */

proc hpfreconcile disaggdata=lv12for
                 aggdata=lv11recfor

```

```

                direction=TD
                outfor=lv12recfor;
            id date interval=month;
            by region product;
run;

/* Verify that level 2 forecasts add up to level 0 forecasts */

proc timeseries data=lv12recfor out=toprec ;
    id date interval=month notsorted accumulate=total;
    var predict;
run;

proc compare base=lv10for 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=lv11for
                aggdata=lv10for
                direction=BU
                outfor=lv10recfor;
    id date interval=month;
    by region;
run;

/* Reconcile forecasts top down from region to region/product */

proc hpfreconcile disaggdata=lv12for
                aggdata=lv11for
                direction=TD
                outfor=lv12recfor;
    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 `lv10fest` 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 hpfxmspec modelrepository=work.mycat
               specname=lv10exm;
  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=mymodels
               selectname=lv10selexm;
  spec lv10exm/ exmmap(predict=predict lower=lower
                       upper=upper stderr=std);
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 lv10recfor and the model specifications contained in the external model lv10exm.

```
/* Create OUTSTAT= and OUTSUM= data sets */

proc hpfengine data=lv10recfor(rename=(actual=sales))
               out=_NULL_
               outstat=lv10outstat
               outsum=lv10outsum
               modelrepository=mymodels
               globalselection=lv10selexm;
  id date interval=month notsorted;
  forecast sales;
  external predict lower
             upper std;
run;
```

---

## Example 9.2: 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 9.1](#) are aggregated to find forecasts at level 1 for the SASHELP.PRICEDATA data set.

```
/* Aggregate region/product forecasts to region level */
```

```

proc hpfreconcile disagdata=lv12for
                direction=BU
                outfor=lv11aggfor
                stdmethod=disagg
                clmethod=gauss;
    id date interval=month;
    by region product;
    aggby region;
run;

```

---

### Example 9.3: 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=lv11sales ;
    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 lv12prop;
    merge tmp lv11sales(rename=(sale=totsale));
    by region date;
    prop = sale / totsale;
run;

/* Compute median sale proportions */

proc sort data=lv12prop;
    by region product;
run;

proc univariate data=lv12prop noprint;

```

```

var prop;
by region product;
output out=lv12medprop median=medprop;
run;

/* Merge median proportions with level2 OUTFOR */

data lv12medfor;
merge lv12for lv12medprop;
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=lv12medfor
                 aggdata=lv11for
                 direction=TD
                 stdmethod=unchanged
                 clmethod=gauss
                 outfor=lv12recmedfor;
disaggdata predict=medprop;
by region product;
run;

```

The variable medprop in the OUTFOR=lv12recmedfor 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=lv12medfor
                 aggdata=lv11for
                 direction=TD
                 predictonly
                 outfor=lv12recmedfor;
disaggdata predict=medprop;
by region product;
run;

```



## Example 9.4: 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 9.1](#) 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
  ;
```

Then you reconcile the two levels by using the SIGN=NONNEGATIVE option to impose the non-negativity 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=lv12for
                 aggdata=lv11for
                 direction=TD
                 sign=nonnegative
                 constraint=constraint
                 outfor=lv12recfor
                 predictonly;
  id date interval=month;
  by region product;
run;
```



# Chapter 10

## The HPFSELECT Procedure

### Contents

---

Overview: HPFSELECT Procedure . . . . .	281
Getting Started: HPFSELECT Procedure . . . . .	282
Syntax: HPFSELECT Procedure . . . . .	283
Functional Summary . . . . .	283
PROC HPFSELECT Statement . . . . .	284
DELETE Statement . . . . .	285
DIAGNOSE Statement . . . . .	285
FORECASTOPTIONS Statement . . . . .	286
SELECT Statement . . . . .	286
SPECIFICATION Statement . . . . .	288
Details: HPFSELECT Procedure . . . . .	292
Examples: HPFSELECT Procedure . . . . .	292
Example 10.1: The INPUTMAP Option . . . . .	292
Example 10.2: The EVENTMAP Option . . . . .	293
Example 10.3: The DIAGNOSE Statement . . . . .	295
Example 10.4: External Models and User-Defined Subroutines . . . . .	298
Example 10.5: Comparing Forecasts from Multiple External Sources . . . . .	299
Example 10.6: Input to User-Defined Subroutines . . . . .	301
Example 10.7: Changing an Existing Selection List . . . . .	303

---

---

### Overview: HPFSELECT Procedure

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 the HPFSELECT procedure, you can control which forecasting model or models SAS High-Performance Forecasting software 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 the HPFSELECT procedure, you can also specify other options that control the forecasting model selection process.

---

## Getting Started: HPFSELECT Procedure

The following example shows how to create a model selection list file. Suppose the model repository MYLIB.MYMODELS contains three model specification files (A.XML, B.XML, C.XML), and you want to create a model selection list that will tell the HPFENGINE procedure to automatically select from these models based on the mean absolute percentage error (MAPE). The following SAS statements accomplish this.

```
proc hpfarimaspec repository=mymodels name=a;
    forecast symbol=y p=12 diflist=(1 12) noint;
    estimate method=ml;
run;

proc hpfesmspec repository=mymodels name=b;
    esm method=winters;
run;

proc hpfucmspec repository=mymodels name=c;
    forecast symbol=y;
    irregular;
    level;
    slope;
    season length=12;
run;

proc hpfselect repository=mymodels
               name=myselect;
    spec a b c;
    select criterion=mape;
run;
```

The new selection list is available for use by the HPFENGINE procedure, shown in the following SAS statements. The GLOBALSELECTION= option references the selection list by name. Selection results are shown in [Figure 10.1](#).

```
proc hpfengine data=sashelp.air
               repository=mymodels
               globalselection=myselect
               print=select
               out=_null_;
    id date interval=month;
    forecast air;
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 the catalog MYLIB.MYMODELS, and the NAME= option specifies that the name of the file be “myselect.xml”. The SPEC statement specifies the list of candidate models. The SELECT statement specifies options that control how the HPFENGINE procedure selects from the candidate models

when applying the selection list MYSELECT to actual time series data.

**Figure 10.1** Selection Results

The HPFENGINE Procedure			
Model Selection Criterion = MAPE			
Model	Statistic	Selected	Label
A	3.1008419	No	ARIMA: Y ~ P = 12 D = (1,12) NOINT
B	3.0845016	Yes	Winters Method (Multiplicative)
C	4.3672632	No	UCM: Y = TREND + SEASON + ERROR

## Syntax: HPFSELECT Procedure

The following statements are used with the HPFSELECT procedure:

```

PROC HPFSELECT options ;
  DELETE specification-list ;
  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.

Statement	Description	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=

Table 10.1 continued

Statement	Description	Option
specifies the forecast holdout sample size as a percentage	SELECT	HOLDOUTPCT=
specifies the model selection criterion	SELECT	CRITERION=
<b>Model Specification Options</b>		
maps specification symbol to data set variable	SPECIFICATION	INPUTMAP
adds event to specification	SPECIFICATION	EVENTAP
associates external data with specification	SPECIFICATION	EXMMAP
associates external subroutine with specification	SPECIFICATION	EXMFUNC
overrides specification labels	SPECIFICATION	LABEL
validates model specifications	PROC HPFSELECT	VALIDATE
<b>Model Selection List Options</b>		
removes specification from the list	DELETE	
specifies an input selection list	PROC HPFSELECT	INSELECTNAME=
<b>Diagnostic Options</b>		
specifies the base value for an intermittent series	DIAGNOSE	IDMBASE=
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.

#### **INSELECTNAME=** *SAS-catalog-name*

provides a selection list as input to the HPFSELECT procedure. The input selection list is specified as a three-level name. The INSELECTNAME= option can be used to add models to existing lists, remove models from existing lists, change selection options, and so forth.

#### **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=.

**VALIDATE**

checks that model specifications exist in the model repository specified in REPOSITORY=.

## DELETE Statement

**DELETE** *specification-list* ;

The DELETE statement is used to remove model specifications from a selection list. There can be any number of model specifications listed in a DELETE statement and any number of DELETE statements.

## DIAGNOSE Statement

**DIAGNOSE** *options* ;

The DIAGNOSE statement is used to specify diagnostic options. DIAGNOSE options are used by the HPFENGINE procedure to subset a model selection list according to certain properties of a time series.

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);
```

**IDMBASE=** *AUTO* | *number*

specifies the base value of the time series used to determine the demand series components for an intermittent demand model. 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 IDMBASE=AUTO is specified, the time series properties are used to automatically adjust the time series. For the common definition of Croston's method use IDMBASE=0, which defines departures based on zero. The default is IDMBASE=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 IDMBASE=AUTO, the base value is automatically determined by the time series median, minimum, and maximum value and the INTERMITTENT= option value.

**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.

**SEASONTTEST=** *option*

specifies the options related to the seasonality test.

The following values for the SEASONTTEST= 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 the HPFENGINE procedure will use seasonal models to forecast the time series. The default is SEASONTTEST=(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;
```



**CHOOSE=** *specification*

specifies the name of a model specification that will be chosen by the HPFENGINE procedure. By default, HPFENGINE will select the model with the best fit in terms of the statistic set by the CRITERION= option. The CHOOSE= option overrides this automatic selection and causes HPFENGINE to generate forecasts by using the model indicated in the option.

**CRITERION=** *option*

specifies the model selection criterion (statistic of fit) to be used to select from several candidate models. This option is often 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
MASE	Mean Absolute Scaled Error
RSQUARE	R-Square
ADJRSQ	Adjusted R-Square
AADJRSQ	Amemiya's Adjusted R-Square
RWRSQ	Random Walk R-Square
AIC	Akaike Information Criterion
AICC	Finite Sample Corrected AIC
SBC	Schwarz Bayesian Information Criterion
APC	Amemiya's Prediction Criterion
MAPES	Mean Absolute Error Percent of Standard Deviation
MDAPES	Median Absolute Error Percent of Standard Deviation
GMAPES	Geometric Mean Absolute Error Percent of Standard Deviation

**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.

---

## SPECIFICATION Statement

**SPECIFICATION** *specification-list* < / *options* > ;

The **SPECIFICATION** statement is used to list model specifications. There can be any number of models specifications in the list and any number of **SPECIFICATION** statements. The **SPECIFICATION** statement can also be written as **SPEC**.

The following options can be used with the **SPECIFICATION** statement.

**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), 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 the **HPFENGINE** procedure 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:

<code>_STDERR_</code>	Prediction standard error
<code>_LOWER_</code>	Lower confidence limit
<code>_UPPER_</code>	Upper confidence limit

The HPFENGINE procedure can also pass data into the user-defined subroutine by using the following options:

<code>_TIMEID_</code>	Time ID
<code>_SEASON_</code>	Seasonal index
<code>_ACTUAL_</code>	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 10.4](#).

**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.

**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 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, the HPFENGINE procedure attempts to locate variables in its DATA= data set that match the symbols in each specification listed in the selection list.

---

## Details: HPFSELECT Procedure

---

---

## Examples: HPFSELECT Procedure

---

---

### Example 10.1: The INPUTMAP Option

In this example, the HPFUCMSPEC procedure is used to define a UCM specification. The dependent variable is assigned the symbol Y, and an input is assigned the symbol X1. You ultimately want to forecast the series contained in the variable MASONRY with the input ELECTRIC. As demonstrated in the following SAS code, the INPUTMAP option in the HPFSELECT procedure is used to tell the HPFENGINE procedure that MASONRY should replace Y and ELECTRIC should replace X1.

```
proc hpfucmspec repository=mymodels
                name=myucm
                label="My UCM spec";
  dependent symbol=Y;
  irregular;
  level;
  slope;
  season length=12;
  input symbol=X1;
run;

proc hpfselect repository=mymodels
              name=myselect;
  spec myucm /
  inputmap(symbol=Y var=MASONRY)
  inputmap(symbol=X1 var=ELECTRIC);
run;
```

The following call to the HPFENGINE procedure creates forecasts using the UCM model with correct variable mappings. The model selection table created is shown in [Output 10.1.1](#).

```
proc hpfengine data=sashelp.workers
              out=_null_
              repository=mymodels
              globalselection=myselect
              print=select;
  id date interval=month;
  forecast masonry;
  stochastic electric;
run;
```

**Output 10.1.1** Selection Results

The HPFENGINE Procedure			
Model Selection Criterion = MAPE			
Model	Statistic	Selected	Label
MYUCM	2.0744810	Yes	My UCM spec

As shown in the following SAS statements, 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. No INPUTMAP option is required.

```
proc hpfucmspec repository=mymodels
    name=myucm
    label="My UCM spec";
    dependent symbol=MASONRY;
    irregular;
    level;
    slope;
    season length=12;
    input symbol=ELECTRIC;
run;

proc hpfselect repository=mymodels
    name=myselect
    label="My Selection List";
    spec myucm;
run;
```

The disadvantage here is that the model specification and data set are tightly linked.

---

## Example 10.2: The EVENTMAP Option

Events are dynamically added as simple regressors to UCM and ARIMA models by using the EVENTMAP option in the SPECIFICATIONS statement.

In this example, 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 following SAS statements illustrate this process. The results are shown in [Output 10.2.1](#).

```
proc hpfevents data=sashelp.air;
    eventdef summer = (june july august);
    eventdata out=eventDB;
run;

proc hpfarimaspec repository=sasuser.repository name=arima;
```

```

forecast symbol=air q=(1 12) transform=log;
run;

proc hpfsselect repository=sasuser.repository name=select;
spec arima;
spec arima / eventmap(symbol=_none_ event=summer);
run;

proc hpfsengine data=sashelp.air
repository=sasuser.repository
outest=outest1
globalselection=select
print=(select estimates)
inevent=eventDB;
id date interval=month;
forecast air;
run;

```

### Output 10.2.1 Selection and Estimation Results

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 for ARIMA Model					
Component	Parameter	Estimate	Standard Error	t Value	Approx 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

In the following statements, 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 10.2.2](#), are the same as the results of the simpler usage with the EVENTMAP option.

```

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 repository=sasuser.repository name=arimasummer;
  forecast symbol=air q=(1 12) transform=log;
  input symbol=summer;
run;

proc hpfselect repository=sasuser.repository name=select;
  spec arima arimasummer;
run;

proc hpfengine data=air
  repository=sasuser.repository
  outest=outest2
  globalselection=select
  print=(select estimates);
  id date interval=month;
  forecast air;
  input summer;
run;

```

### Output 10.2.2 Selection and Estimation Results

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 for ARIMASUMMER Model					
Component	Parameter	Estimate	Standard Error	t Value	Approx 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 10.3: The DIAGNOSE Statement

The DIAGNOSE statement enables control of the diagnostics used by the HPFENGINE procedure 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 the HPFENGINE procedure judge the dependent series as seasonal and therefore exclude the nonseasonal model from consideration. The following statements illustrate the behavior with the explicit use of the DIAGNOSE statement. The results are shown in [Output 10.3.1](#).

```
proc hpfesmspec repository=sasuser.repository name=logdouble;
    esm method=double transform=log;
run;

proc hpfesmspec repository=sasuser.repository name=logwinters;
    esm method=winters transform=log;
run;

proc hpfselect repository=sasuser.repository name=select;
    spec logdouble logwinters;
run;

proc hpfengine data=sashelp.air
    repository=sasuser.repository
    outest=outest1
    globalselection=select
    print=all;
    id date interval=month;
    forecast air;
run;
```

**Output 10.3.1** Seasonality Test at Significance Level of 0.01

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)

In the following statements, 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 results are shown in [Output 10.3.2](#).

```
proc hpfselect repository=sasuser.repository name=select;
    spec logdouble logwinters;
    diagnose seastest=none;
run;

proc hpfengine data=sashelp.air
    repository=sasuser.repository
    outest=outest1
    globalselection=select print=select;
```

```

id date interval=month;
forecast air;
run;

```

### Output 10.3.2 No Seasonality Test Performed

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, in the following statements, the seasonality test significance level is set to zero so that the series will not be judged as seasonal. In [Output 10.3.3](#), note that the nonseasonal model is fit to the series, but the seasonal model is removed.

```

proc hpfselect repository=sasuser.repository name=select;
  spec logdouble logwinters;
  diagnose seastest=(siglevel=0);
run;

proc hpfengine data=sashelp.air
  repository=sasuser.repository
  outest=outest1
  globalselection=select
  print=select;
  id date interval=month;
  forecast air;
run;

```

### Output 10.3.3 All Series Treated as Nonseasonal

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 10.4: External Models and User-Defined Subroutines

The EXMFUNC option enables you to associate an external model specification with a user-defined subroutine. In this example, you define a user subroutine and store it in a catalog. The following statements create a simple three-point moving average.

```
proc fcmp outlib=work.hpfengine.funcs;
  subroutine move_avg3(act[*], pred[*]);
    outargs 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=work.hpfengine;
```

Now, just for comparison, you use the HPFARIMASPEC procedure to make a model specification that will produce the same three-point moving average.

```
proc hpfarimaspec repository=sasuser.repository name=arima;
  forecast symbol=y noint p=3
            ar=(0.333333333 0.333333333 0.333333333);
  estimate noest;
run;
```

Next, you 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 repository=sasuser.repository name=myexm;
  exm;
run;

proc hpfselect repository=sasuser.repository name=select;
  diagnose seautest=none;
  spec arima;
  spec myexm / exmfunc('move_avg3(_actual_ _predict_ )')
              label="External Model from move_avg3";
run;
```

Finally, you 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_
    repository=sasuser.repository
    globalselection=select
    print=select;
    id date interval=month;
    forecast air;
run;
```

The output is shown in [Output 10.4.1](#).

#### Output 10.4.1 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 10.5: 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 these 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. The following statements generate external data.

```
data temp;
    set sashelp.air;
    drop i;
    * introduce some missing for EXPAND to fill in;
    if mod(_n_, 5) eq 0 then air = .;
    if mod(_n+1, 5) eq 0 then air = .;
    if mod(_n+2, 5) eq 0 then 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;
run;

data arimaout;
    set arimaout;
    airline = exp(airline);
run;

data temp;
    keep date interp airline air;
    merge expandout arimaout sashelp.air;
    by date;
run;

```

Next, you 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 repository=sasuser.repository name=esm;
    esm method=seasonal transform=auto;
run;

proc hpfexmspec repository=sasuser.repository name=exm;
    exm;
run;

proc hpfselect repository=sasuser.repository 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_
      repository=sasuser.repository
      globalselection=select
      lead=4
      print=select;
id date interval=month;
forecast air;
external interp airline;
run;

```

The output is shown in [Output 10.5.1](#).

### Output 10.5.1 Two External Forecasts and a Smoothing Model

The HPFENGINE Procedure			
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 10.6: Input to User-Defined Subroutines

This example illustrates the different type of data that the HPFENGINE procedure can pass to a user-defined subroutine. Consider the following subroutine definition.

```

proc fcmp outlib=work.hpfengine.funcs;
  subroutine testsub(timeid[*], act[*], seasons[*], pred[*]);
    outargs pred;
    actlen = DIM(act);
    predlen = DIM(pred);
    format date monyy.;

    * print the input;
    do i=6 to 18;
      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;
  endsub;
run;

```

```

        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 hpfexmspec repository=sasuser.repository name=myexml;
    exm;
run;

proc hpfselect repository=sasuser.repository name=select;
    diagnose seastest=none;
    spec myexml / exmfunc('testsub(_timeid_ _actual_ _season_ _predict_)');
run;

```

Then the following call to the HPFENGINE procedure invokes the user-defined subroutine named TESTSUB.

```

options cmlib = work.hpfengine;
proc hpfengine data=sashelp.air
    out=_null_
    outfor=outfor
    repository=sasuser.repository
    globalselection=select;
    id date interval=month;
    forecast air;
run;

```

A partial listing of the log output from the run of the HPFENGINE procedure follows. The seasonal cycle length is 12, and thus the zero-based seasonal index repeats 0 through 11. Though not used in this simple subroutine, all these data are available when you are computing forecasts.

```

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

```



## Example 10.7: Changing an Existing Selection List

The following SAS statements delete three specifications from a copy of SASHELP.HPFDFLT.BEST and add a new user-defined specification. The resulting list is named SASUSER.REPOSITORY.SELECT.

```
proc hpfarimaspec repository=sasuser.repository name=arima;
    forecast symbol=y q=(1 12) dif=(1 12) noint transform=log;
run;

proc hpfselect name=select repository=sasuser.repository
    inselectname=sashelp.hpfdflt.best;
    delete smsimp smdamp smwint;
    spec arima;
run;
```

The CATNAME statement that follows gives the HPFENGINE procedure access to the model specifications in both catalogs SASHELP.HPFDFLT and SASUSER.REPOSITORY. The HPFENGINE produces the output shown in [Output 10.7.1](#).

```
catname twocats (sashelp.hpfdflt sasuser.repository);

proc hpfengine data=sashelp.air repository=work.twocats
    globalselection=select print=select out=_null_;
    id date interval=month;
    forecast air;
run;
```

### Output 10.7.1 Modified Selection List

The HPFENGINE Procedure					
Model Selection Criterion = MAPE					
Model	Statistic	Selected	Label		
ARIMA	2.9672282	Yes	ARIMA: Log( Y ) ~ D = (1,12) Q = (1,12) NOINT		
smdoub	.	Removed	Double Exponential Smoothing		
smlin	.	Removed	Linear Exponential Smoothing		
smadwn	3.5343216	No	Winters Method (Additive)		
smseas	3.5196140	No	Seasonal Exponential Smoothing		



# Chapter 11

## The HPFUCMSPEC Procedure

### Contents

---

Overview . . . . .	<b>305</b>
Getting Started . . . . .	<b>306</b>
Syntax . . . . .	<b>306</b>
Functional Summary . . . . .	307
PROC HPFUCMSPEC Statement . . . . .	309
AUTOREG Statement . . . . .	309
BLOCKSEASON Statement . . . . .	310
CYCLE Statement . . . . .	311
DEPLAG Statement . . . . .	313
FORECAST Statement . . . . .	314
INPUT Statement . . . . .	315
IRREGULAR Statement . . . . .	316
LEVEL Statement . . . . .	316
SEASON Statement . . . . .	317
SLOPE Statement . . . . .	319
Examples . . . . .	<b>319</b>
Example 11.1: Some Syntax Illustrations . . . . .	319
Example 11.2: How to Include a UCM Model in a Model Selection List . . . . .	321
Example 11.3: How to Create a Generic Seasonal Model Spec That Is Suitable for Different Season Lengths . . . . .	323
References . . . . .	<b>325</b>

---

---

### 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 4, “[The HPFENGINE Procedure](#),”). You can specify any UCM model that can be analyzed using the UCM procedure; see Chapter 28, “[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=mymodels
    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 MYMODELS, 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
<b>Model Repository Options</b>		
specify the model repository	PROC HPFUCMSPEC	REPOSITORY=
specify the model specification name	PROC HPFUCMSPEC	NAME=
specify the model specification label	PROC HPFUCMSPEC	LABEL=
<b>Options for Specifying Symbolic Series Names</b>		
specify a symbolic name for the response series	FORECAST	SYMBOL=
specify a symbolic name for the input series	INPUT	SYMBOL=
<b>Options for Specifying the Model</b>		
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=

Description	Statement	Option
fix the value of the disturbance variance of the level component to the specified initial value	LEVEL	NOEST
specify the initial value for the disturbance variance of the slope component	SLOPE	VARIANCE=
fix the value of the disturbance variance of the slope component to the specified initial value	SLOPE	NOEST
specify the season length of a seasonal component	SEASON	LENGTH=
specify the type of a seasonal component	SEASON	TYPE=
specify the initial value for the disturbance variance of a seasonal component	SEASON	VARIANCE=
fix the value of the disturbance variance of the seasonal component to the specified initial value	SEASON	NOEST
specify the block size of a block seasonal component	BLOCKSEASON	BLOCKSIZE=
specify the number of blocks of a block seasonal component	BLOCKSEASON	NBLOCKS=
specify the relative position of the first observation within the block of a block seasonal component	BLOCKSEASON	OFFSET=
specify the initial value for the disturbance variance of a block seasonal component	BLOCKSEASON	VARIANCE=
fix the value of the disturbance variance of the block seasonal component to the specified initial value	BLOCKSEASON	NOEST
specify the initial value for the period of a cycle component	CYCLE	PERIOD=
specify the initial value for the damping factor of a cycle component	CYCLE	RHO=
specify the initial value for the disturbance variance of the cycle component	CYCLE	VARIANCE=
fix the values of the parameters of the cycle component to the specified initial values	CYCLE	NOEST=
specify the initial value for the damping factor of the autoreg component	AUTOREG	RHO=
specify the initial value for the disturbance variance of the autoreg component	AUTOREG	VARIANCE=
fix the values of the parameters of the autoreg component to the specified initial values	AUTOREG	NOEST=

Description	Statement	Option
specify the lags of the response series to be included in the model	DEPLAG	LAGS=
specify the initial values for the lag coefficients for the response lags	DEPLAG	PHI=
fix the values of lag coefficients to the specified initial values	DEPLAG	NOEST

## 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  $\pi$ . It is modeled separately for easier interpretation. A stochastic equation for an autoregressive component  $r_t$  can be written as follows:

$$r_t = \rho r_{t-1} + v_t, \quad v_t \sim i.i.d. N(0, \sigma_v^2)$$

The damping factor  $\rho$  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  $\rho$  and the disturbance variance  $\sigma_v^2$  are estimated from the data.

**NOEST=RHO**

**NOEST= VARIANCE**

**NOEST=( RHO VARIANCE )**

This option fixes the values of  $\rho$  and  $\sigma_v^2$  to those specified in RHO= and VARIANCE= options.

**RHO= value**

This option is used to supply an initial value for the damping factor  $\rho$  during the parameter estimation process. The value of  $\rho$  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  $\sigma_v^2$  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  $\gamma_t$  that has a special block structure. The seasonal  $\gamma_t$  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  $\tau_1$ , the next  $m$  effects are all equal to some number  $\tau_2$ , 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,  $\sigma_{\omega}^2$ , in the  $\gamma_t$  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,  $\psi_t$ , in the model. The stochastic equation governing a cycle component of period  $p$  and damping factor  $\rho$  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  $\nu_t$  and  $\nu_t^*$  are independent, zero mean, Gaussian disturbances with variance  $\sigma_\nu^2$  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  $\rho$  can be any value in the interval (0, 1), including 1 but excluding 0. The cycles with the frequency zero and  $\pi$ , which correspond to the periods equal to infinity and two respectively, can be specified using the AUTOREG statement. The values of  $\rho$  smaller than 1 give rise to a stationary cycle, while  $\rho = 1$  gives rise to a nonstationary cycle. As a default, values of  $\rho$ ,  $p$ , and  $\sigma_\nu^2$  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  $\sigma_\nu^2$  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  $\sigma_\nu^2$ .

```
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,  $\sigma_v^2$ , 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  $\phi_1$  and  $\phi_2$  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  $\phi_1$  and  $\phi_2$  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<sub>1</sub> > ... (lag, ..., lag) < s<sub>k</sub> >**

This is a required option in this statement. LAGS=( $l_1, l_2, \dots, l_k$ ) 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( <i>n</i> )	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( <i>n</i> )	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,  $\epsilon_t$ , in the model; it is modeled as a sequence of independent, zero mean, Gaussian random variables with variance  $\sigma_\epsilon^2$ . The options in this statement enable you to specify the value of  $\sigma_\epsilon^2$  and to output the forecasts of  $\epsilon_t$ . As a default,  $\sigma_\epsilon^2$  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  $\sigma_\epsilon^2$ , to be used in the nonlinear parameter estimation process.

```
irregular variance=4;
```

### NOEST

This option fixes the value of  $\sigma_\epsilon^2$  to the value specified in the VARIANCE= option.

### VARIANCE= value

This option is used to supply an initial value for  $\sigma_\epsilon^2$  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,  $\mu_t$ , 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” on page 319), a Locally Linear Trend (LLT) is obtained. The equations of LLT are as follows:

$$\begin{aligned} \mu_t &= \mu_{t-1} + \beta_{t-1} + \eta_t, & \eta_t &\sim i.i.d. N(0, \sigma_\eta^2) \\ \beta_t &= \beta_{t-1} + \xi_t, & \xi_t &\sim i.i.d. N(0, \sigma_\xi^2) \end{aligned}$$

In either case, the options in the LEVEL statement are used to specify the value of  $\sigma_\eta^2$  and to request forecasts of  $\mu_t$ . The SLOPE statement is used for similar purposes in the case of slope  $\beta_t$ . 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  $\sigma_\eta^2$  fixed at 4. The value of  $\sigma_\xi^2$ , 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  $\sigma_\eta^2$  to the value specified in the VARIANCE= option.

## VARIANCE= value

This option is used to supply an initial value for  $\sigma_\eta^2$ , the disturbance variance in the  $\mu_t$  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,  $\gamma_t$ , 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:

$$\sum_{i=0}^{s-1} \gamma_{t-i} = \omega_t, \quad \omega_t \sim i.i.d. N(0, \sigma_\omega^2)$$

The equations for a TRIGONOMETRIC type seasonal are as follows:

$$\gamma_t = \sum_{j=1}^{\lfloor s/2 \rfloor} \gamma_{j,t}$$

where  $\lfloor s/2 \rfloor$  equals  $s/2$  if  $s$  is even and equals  $(s - 1)/2$  if it is odd. The sinusoids  $\gamma_{j,t}$  have frequencies  $\lambda_j = 2\pi j/s$  and are specified by the matrix equation

$$\begin{bmatrix} \gamma_{j,t} \\ \gamma_{j,t}^* \end{bmatrix} = \begin{bmatrix} \cos \lambda_j & \sin \lambda_j \\ -\sin \lambda_j & \cos \lambda_j \end{bmatrix} \begin{bmatrix} \gamma_{j,t-1} \\ \gamma_{j,t-1}^* \end{bmatrix} + \begin{bmatrix} \omega_{j,t} \\ \omega_{j,t}^* \end{bmatrix}$$

where the disturbances  $\omega_{j,t}$  and  $\omega_{j,t}^*$  are assumed to be independent and, for fixed  $j$ ,  $\omega_{j,t}$  and  $\omega_{j,t}^* \sim N(0, \sigma_\omega^2)$ . If  $s$  is even then the equation for  $\gamma_{s/2,t}^*$  is not needed and  $\gamma_{s/2,t}$  is given by

$$\gamma_{s/2,t} = -\gamma_{s/2,t-1} + \omega_{s/2,t}$$

Note that, whether the seasonal type is DUMMY or TRIGONOMETRIC, there is only one parameter, the disturbance variance  $\sigma_\omega^2$ , 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  $\sigma_\omega^2$  is estimated from the data. The following statement specifies a trigonometric seasonal with monthly seasonality. It also provides a starting value for  $\sigma_\omega^2$ .

```
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 11.3](#). 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,  $\sigma_\omega^2$ , in the  $\gamma_t$  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  $\mu_t$  and slope  $\beta_t$ , are as follows:

$$\begin{aligned}\mu_t &= \mu_{t-1} + \beta_{t-1} + \eta_t, & \eta_t &\sim i.i.d. N(0, \sigma_\eta^2) \\ \beta_t &= \beta_{t-1} + \xi_t, & \xi_t &\sim i.i.d. N(0, \sigma_\xi^2)\end{aligned}$$

The SLOPE statement is used to specify the value of the disturbance variance,  $\sigma_\xi^2$ , in the slope equation, and to request forecasts of  $\beta_t$ . The following examples illustrate this statement:

```
level;
slope;
```

These statements request that a locally linear trend be used in the model. The disturbance variances  $\sigma_\eta^2$  and  $\sigma_\xi^2$  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,  $\sigma_\xi^2$ , to the value specified in the VARIANCE= option.

### VARIANCE= value

This option is used to supply an initial value for the disturbance variance,  $\sigma_\xi^2$ , in the  $\beta_t$  equation, at the start of the parameter estimation process. Any nonnegative value, including zero, is an acceptable starting value.

---

## Examples

---

### Example 11.1: 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, `work.mymodels`, and the models are named as *modell1*, *modell2*, ... 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=mymodels
    name=modell1;
    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=mymodels
    name=modell2;
    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=mymodels
    name=modell3;
    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=mymodels
    name=modell4;
    forecast symbol=sales transform=log;
    irregular;
    level;
    slope variance=10;
    cycle period=12 rho=0.9 noest=(period);
    cycle period=6 noest=(period);
run;

title "Models Added to MYMODELS Repository";

```

```
proc catalog catalog=mymodels;
  contents;
run;
```

#### Output 11.1.1 Listing of Models in MYMODELS repository

Models Added to MYMODELS Repository				
Contents of Catalog WORK.MYMODELS				
#	Name	Type	Create Date	Modified Date Description
1	BSM1	XML	09Jul07:16:40:11	09Jul07:16:40:11 Basic structural model with one input
2	MODEL1	XML	09Jul07:16:40:11	09Jul07:16:40:11
3	MODEL2	XML	09Jul07:16:40:11	09Jul07:16:40:11
4	MODEL3	XML	09Jul07:16:40:11	09Jul07:16:40:11
5	MODEL4	XML	09Jul07:16:40:11	09Jul07:16:40:11

## Example 11.2: 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 4, “[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 10, “[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
... more lines ...
```

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.

```
proc hpfucmspec repository=mymodels
    name=m1;
    forecast symbol=y;
    irregular;
    level;
run;

proc hpfucmspec repository=mymodels
    name=m2;
    forecast symbol=y;
    irregular;
    level;
    input symbol=x1;
run;

proc hpfucmspec repository=mymodels
    name=m3;
    forecast symbol=y;
    irregular;
    level;
    input symbol=x1;
    input symbol=x2;
    input symbol=x3;
run;
```

The follow statements create a selection list that includes model specifications  $m1$ ,  $m2$  and  $m3$ .

```
proc hpfselect repository=mymodels
    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 Figure 11.2.1).

```
proc hpfengine data=nile
    repository=mymodels
    globalselection=myselect
    lead=0
    print=(select);
forecast riverFlow;
input Shift1899;
input Event1877;
input Event1913;
run;
```

### Output 11.2.1 Model Selection Based on the MAPE Criterion

Models Added to MYMODELS Repository			
The HPFENGINE Procedure			
Model Selection Criterion = MAPE			
Model	Statistic	Selected	Label
M1	13.096557	No	UCM: Y = LEVEL + ERROR
M2	11.978729	No	UCM: Y = LEVEL + X1 + ERROR
M3	10.532463	Yes	UCM: Y = LEVEL + X1 + ... + X3 + ERROR

## Example 11.3: 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 statements.

```
proc hpfucmspec repository=mymodels
    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=mymodels
                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 [Figure 11.3.1](#) and [Figure 11.3.2](#), respectively.

```
/* Create a selection list that contains
   the Generic Airline Model */
proc hpfselect repository=mymodels
              name=genselect;
spec GenericBSM;
run;

proc hpfengine data=sashelp.air
              repository=mymodels
              globalselection=genselect
              print=(estimates);
id date interval=month;
forecast air;
run;
```

**Output 11.3.1** Parameter Estimates for the Monthly Series

Models Added to MYMODELS Repository					
The HPFENGINE Procedure					
Parameter Estimates for GENERICBSM Model					
Component	Parameter	Estimate	Standard Error	t Value	Approx 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;

proc hpfengine data=Qair
  repository=mymodels
  globalselection=genselect
  print=(estimates);
  id date interval=quarter;
  forecast air;
run;

```

### Output 11.3.2 Parameter Estimates for the Quarterly Series

Models Added to MYMODELS Repository					
The HPFENGINE Procedure					
Parameter Estimates for GENERICBSM Model					
Component	Parameter	Estimate	Standard Error	t Value	Approx Pr >  t
IRREGULAR	ERROR VARIANCE	6.7904E-11	1.32294E-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.





# Chapter 12

## The HPF Procedure

### Contents

---

Overview . . . . .	328
Getting Started . . . . .	329
Syntax . . . . .	332
Functional Summary . . . . .	332
PROC HPF Statement . . . . .	334
BY Statement . . . . .	337
FORECAST Statement . . . . .	338
ID Statement . . . . .	343
IDM Statement . . . . .	346
Smoothing Model Specification Options for IDM Statement . . . . .	348
Details . . . . .	351
Smoothing Model Parameter Specification Options . . . . .	351
Smoothing Model Forecast Bounds Options . . . . .	351
Accumulation . . . . .	352
Missing Value Interpretation . . . . .	353
Diagnostic Tests . . . . .	354
Model Selection . . . . .	354
Transformations . . . . .	354
Parameter Estimation . . . . .	354
Missing Value Modeling Issues . . . . .	355
Forecasting . . . . .	355
Inverse Transformations . . . . .	355
Statistics of Fit . . . . .	355
Forecast Summation . . . . .	356
Comparison to the Time Series Forecasting System . . . . .	356
Data Set Output . . . . .	356
OUT= Data Set . . . . .	356
OUTEST= Data Set . . . . .	357
OUTFOR= Data Set . . . . .	357
OUTPROCINFO= Data Set . . . . .	358
OUTSTAT= Data Set . . . . .	358
OUTSUM= Data Set . . . . .	359
OUTSEASON= Data Set . . . . .	360
OUTTREND= Data Set . . . . .	361

Printed Output . . . . .	362
ODS Table Names . . . . .	363
ODS Graphics . . . . .	364
Examples . . . . .	<b>366</b>
Example 12.1: Automatic Forecasting of Time Series Data . . . . .	366
Example 12.2: Automatic Forecasting of Transactional Data . . . . .	368
Example 12.3: Specifying the Forecasting Model . . . . .	370
Example 12.4: Extending the Independent Variables for Multivariate Forecasts . . . . .	370
Example 12.5: Forecasting Intermittent Time Series Data . . . . .	372
Example 12.6: Illustration of ODS Graphics . . . . .	374
References . . . . .	<b>378</b>

---

## 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 (additive and multiplicative)
  - 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.

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 35, "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 13, "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" on page 366 section.

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 criterion=mape;
  forecast cars / model=bestn criterion=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 criterion=mape;
  forecast cars / model=bestn criterion=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 , HUB-CAPS , 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 o ;
ptions
```

---

## Functional Summary

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

Description	Statement	Option
<b>Statements</b>		
specify BY-group processing	BY	
specify variables to forecast	FORECAST	
specify the time ID variable	ID	
specify intermittent demand model	IDM	
<b>Data Set Options</b>		
specify the input data set	PROC HPF	DATA=
specify to output forecasts only	PROC HPF	NOOUTALL
specify the output data set	PROC HPF	OUT=

Description	Statement	Option
specify parameter output data set	PROC HPF	OUTEST=
specify forecast output data set	PROC HPF	OUTFOR=
specify the forecast procedure information output data set	PROC HPF	OUTPROCINFO=
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
replace missing values	FORECAST	REPLACEMISSING
use forecast value to append	FORECAST	USE=
<b>Accumulation and Seasonality Options</b>		
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=
<b>Forecasting Horizon, Holdout, Holdback Options</b>		
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=
<b>Forecasting Model and Selection Options</b>		
specify confidence limit width	FORECAST	ALPHA=
specify model selection criterion	FORECAST	CRITERION=
specify intermittency	FORECAST	INTERMITTENT=
specify forecast model	FORECAST	MODEL=
specify median forecasts	FORECAST	MEDIAN
specify backcast initialization	FORECAST	NBACKCAST=
specify seasonality test	FORECAST	SEASONTEST=
specify model transformation	FORECAST	TRANSFORM=
<b>Intermittent Demand Model Options</b>		
specify model for average demand	IDM	AVERAGE=
specify the base value	IDM	BASE=

Description	Statement	Option
specify model for demand intervals	IDM	INTERVAL=
specify model for demand sizes	IDM	SIZE=
Printing and Plotting Control Options		
specify time ID format	ID	FORMAT=
specify graphical output	PROC HPF	PLOT=
specify printed output	PROC HPF	PRINT=
specify detailed printed output	PROC HPF	PRINTDETAILS
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.

**OUTPROCINFO=** *SAS-data-set*

names the output data set to contain information in the SAS log, specifically the number of notes, errors, and warnings and the number of series processed, forecasts requested, and forecasts failed.

**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.

**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.
MODELFORECASTONLY	plots forecast graphics with confidence limits in the data range.
FORECASTSONLY	plots the forecast in the forecast horizon 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.

**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.

**CRITERION=** *option*

**SELECT=** *option*

specifies the model selection criterion (statistic of fit) to be used to select from several candidate models. This option is often used in conjunction with the HOLDOUT= option. The CRITERION= option can also be specified as SELECT=. The default is CRITERION=RMSE.

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 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

**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 CRITERION= 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 `CRITERION=` 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 <code>OUT=</code> data set. This option is particularly useful when the results stored in the <code>OUT=</code> 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
ADDSEASONALISEASONAL	Additive Seasonal Exponential Smoothing

MULTSEASONAL	Multiplicative Seasonal Exponential Smoothing
WINTERS	Winters Multiplicative Method
ADDWINTERS	Winters Additive Method
BEST	Best Candidate Smoothing Model (SIMPLE, DOUBLE, LINEAR, DAMPTREND, ADDSEASONAL, WINTERS, ADDWINTERS)
BESTN	Best Candidate Nonseasonal Smoothing Model (SIMPLE, DOUBLE, LINEAR, DAMPTREND)
BESTS	Best Candidate Seasonal Smoothing Model (ADDSEASONAL, WINTERS, ADDWINTERS)
IDMICROSTON	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 HOLD-OUT= option and CRITERION= 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 14, "[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.

#### **SEASONTTEST= *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:

SEASONTTEST=NONE no test

SEASONTTEST=(SIGLEVEL=number) Significance probability value.

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

**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( <i>n</i> )	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.

**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. 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 criterion=mape )
    size      =( method=simple transform=auto criterion=mape );
forecast sales / model=idm criterion=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 criterion=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 criterion=mape)
    size      =(method=simple transform=auto criterion=mape)
    average   =(method=simple transform=auto criterion=mape);
forecast sales / model=idm criterion=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:

**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.

**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.

---

## 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) criterion=rmse bounds=(1, .));

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

average   =(transform=auto method=bestn
          levelrest=(0.0001 0.9999)
          trendrest=(0.0001 0.9999)
          damprest =(0.0001 0.9999) criterion=rmse);
```

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

The following describes the smoothing model options:

**BOUNDS=( number , number )**

Specifies the component forecast bound. See the smoothing model forecast bounds 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.

**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.

**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 criterion= 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)

**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.

**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.

**CRITERION=** *option*

**SELECT=** *option*

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

**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( <i>n</i> )	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.

**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.



---

## Details

---

### 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.

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                 | <code>ACCUMULATE=</code> option  |
| 2. Missing Value Interpretation | <code>SETMISSING=</code> option  |
| 3. Diagnostic Tests             | <code>INTERMITTENT=</code> and <code>SEASONTEST=</code> options  |
| 4. Model Selection              | <code>MODEL=</code> , <code>HOLDOUT=</code> , <code>HOLDOUTPCT=</code> , and <code>CRITERION=</code> options |
| 5. Transformations              | <code>TRANSFORM=</code> option   |
| 6. Parameter Estimation         | <code>MODEL=</code> option   |

7. Forecasting	MODEL= and LEAD= options
8. Inverse Transformation	TRANSFORM= and MEDIAN options
9. Statistics of Fit	CRITERION= 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:

```

01MAR1999    40
01APR1999    .
01MAY1999    90

```

If the ACCUMULATE=AVERAGE option is specified:

```

01MAR1999    20
01APR1999    .
01MAY1999    30

```

If the ACCUMULATE=MINIMUM option is specified:

```
O1MAR1999    10
O1APR1999    .
O1MAY1999    20
```

If the ACCUMULATE=MEDIAN option is specified:

```
O1MAR1999    20
O1APR1999    .
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](#)” on page 355 section). 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 SEASONTTEST= 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 CRITERION= 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 14, “[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:

<code>_NAME_</code>	Variable name
<code>_MODEL_</code>	Forecasting Model
<code>_TRANSFORM_</code>	Transformation
<code>_PARAM_</code>	Parameter Name
<code>_EST_</code>	Parameter Estimate
<code>_STDERR_</code>	Standard Errors
<code>_TVALUE_</code>	<i>t</i> -Values
<code>_PVALUE_</code>	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:

<code>_NAME_</code>	Variable name
<code>_TIMEID_</code>	Time ID values
ACTUAL	Actual 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.

---

## OUTPROCINFO= Data Set

The OUTPROCINFO= data set contains information about the run of the HPF procedure. The following variables are present:

<code>_SOURCE_</code>	Set to the name of the procedure, in this case HPF.
<code>_NAME_</code>	Name of an item being reported; may be the number of errors, notes, or warnings, number of forecasts requested, etc.
<code>_LABEL_</code>	Descriptive label for the item in <code>_NAME_</code> .
<code>_STAGE_</code>	Set to the current stage of the procedure, for HPFENGINE this is set to ALL.
<code>_VALUE_</code>	Value of the item specified in <code>_NAME_</code> .

---

## 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:

<code>_NAME_</code>	Variable name
<code>_REGION_</code>	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:

<code>_NAME_</code>	Variable name
<code>_STATUS_</code>	Forecasting Status. Nonzero values imply that no forecast was generated for the series.
<code>NOBS</code>	Number of Observations
<code>N</code>	Number of Nonmissing Observations
<code>NMISS</code>	Number of Missing Observations
<code>MIN</code>	Minimum Value
<code>MAX</code>	Maximum Value
<code>MEAN</code>	Mean Value
<code>STDDEV</code>	Standard Deviation

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

<code>PREDICT</code>	Forecast Summation Predicted Values
<code>STD</code>	Forecast Summation Prediction Standard Errors
<code>LOWER</code>	Forecast Summation Lower Confidence Limits
<code>UPPER</code>	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:

<code>_LEAD<math>n</math>_</code>	Multistep Forecast ( $n$ ranges from one to <code>LEAD=number</code> ). If <code>USE=LOWER</code> , this variable will contain the lower confidence limits; if <code>USE=UPPER</code> , 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:

<code>_NAME_</code>	Variable name
<code>_TIMEID_</code>	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=PERFORMANCE SUMMARY

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 12.2** 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	

**Table 12.2** (continued)

ODS Table Name	Description	Option
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	Evaluation Statistics of Fit	
PerformanceStatistics	Performance (Out-of-Sample)	
	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

This section describes the use of ODS for creating graphics with the HPF procedure. To request these graphs, you must specify the ODS GRAPHICS statement. In addition, you can specify the PLOT= option in the HPF statement.

### 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 12.3](#).

To request these graphs, you must specify the ODS GRAPHICS statement. In addition, you can specify the **PLOT=** option in the HPF statement.

**Table 12.3** ODS Graphics Produced by PROC HPF

<b>ODS Graph Name</b>	<b>Plot Description</b>	<b>Statement</b>	<b>PLOT= Option</b>
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

**Table 12.3** (continued)

ODS Graph Name	Plot Description	Statement	Option
ModelPlot	Model Plot	PROC HPF	PLOT=MODELS
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 12.1: 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;
... more lines ...
```

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 SGPLOT procedure statements plot the forecasts related to shoe sales:



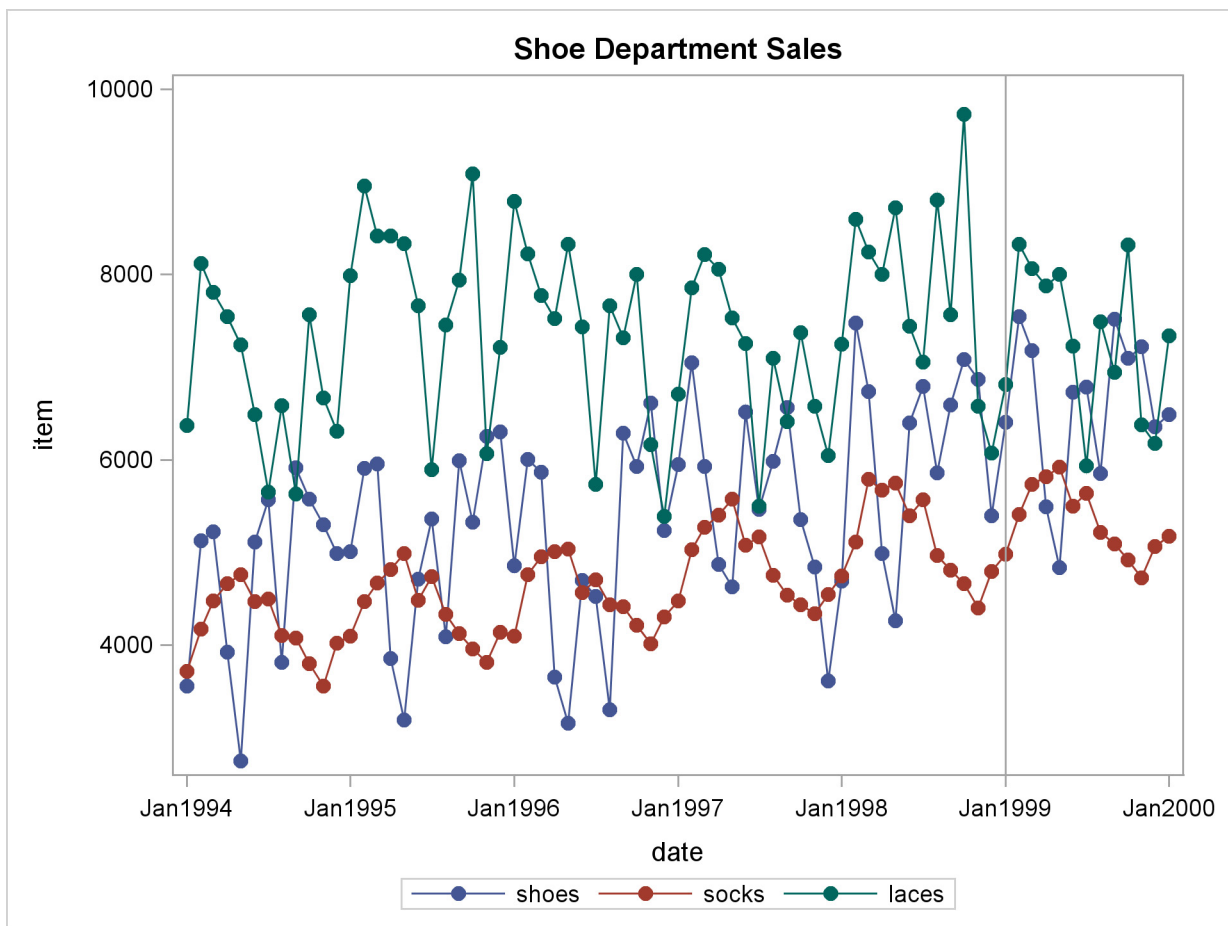
```

title "Shoe Department Sales";
proc sgplot data=nextyear;
  series x=date y=shoes / markers
  markerattrs=(symbol=circlefilled) lineattrs=(pattern=1);
  series x=date y=socks / markers
  markerattrs=(symbol=circlefilled) lineattrs=(pattern=1);
  series x=date y=laces / markers
  markerattrs=(symbol=circlefilled) lineattrs=(pattern=1);
  xaxis values=('01JAN1994'd to '01DEC2000'd by year);
  yaxis label='item';
  refline '01JAN1999'd / axis=x;
run;

```

The GPLOT procedure results are shown in Figure 12.1.1. 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 12.1.1 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 12.1.2](#). This output shows the predictions, prediction standard errors and the upper and lower confidence limits for the next twelve monthly periods.

### Output 12.1.2 Forecast Tables

Shoe Department Sales					
The HPF Procedure					
Forecasts for Variable shoes					
Obs	Time	Forecasts	Standard 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 12.2: 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 data set WEBSITES contains data recorded at several Internet Web sites. WEBSITES contains a variable 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.

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 GPROT procedure statements plot the forecasts related to the Internet data:

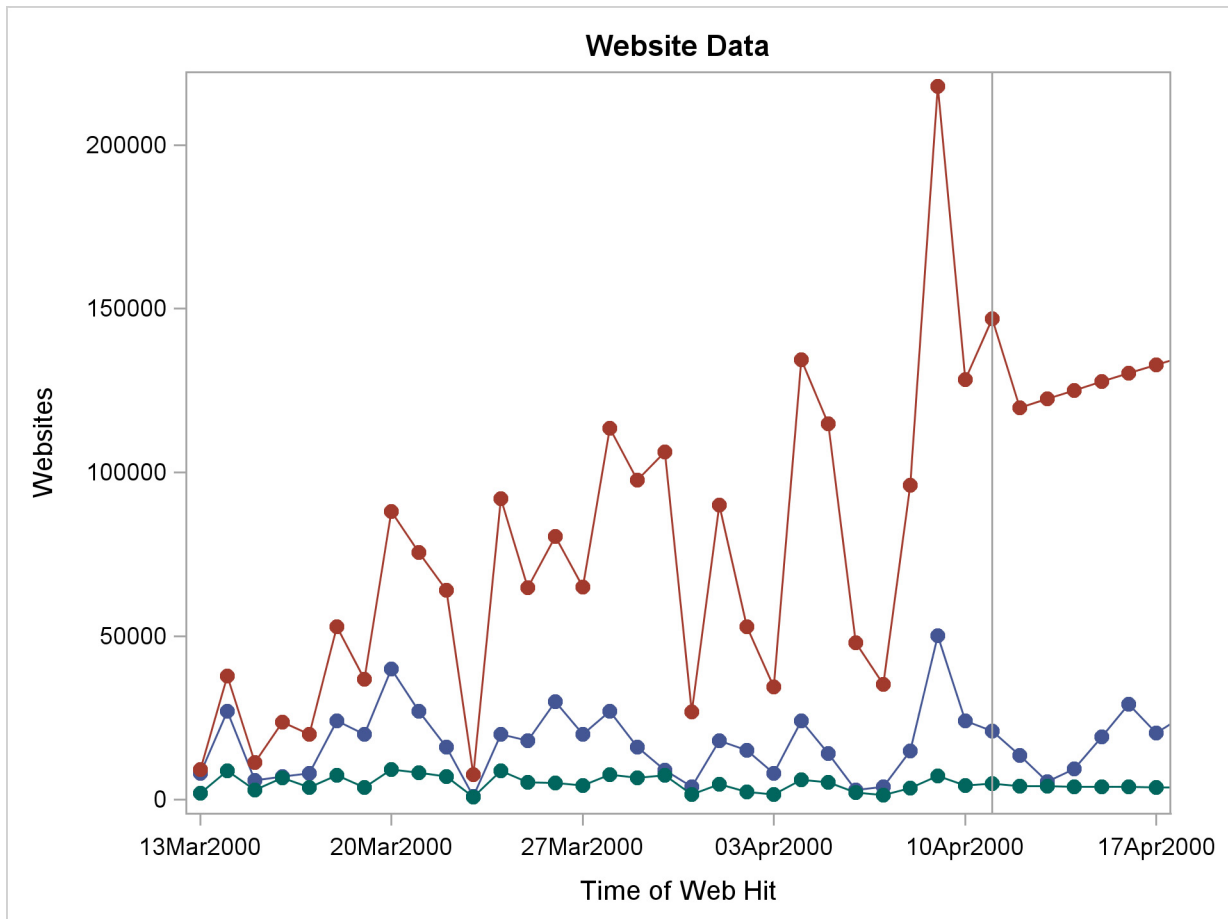
```

title1 "Website Data";
proc sgplot data=nextweek noautolegend;
  series x=time y=boats / markers
  markerattrs=(symbol=circlefilled) lineattrs=(pattern=1);
  series x=time y=cars / markers
  markerattrs=(symbol=circlefilled) lineattrs=(pattern=1);
  series x=time y=planes / markers
  markerattrs=(symbol=circlefilled) lineattrs=(pattern=1);
  xaxis values=('13MAR2000:00:00:00'dt to '18APR2000:00:00:00'dt by dtweek);
  yaxis label='Websites';
  refline '11APR2000:00:00:00'dt / axis=x;
run;

```

The GPROT procedure results are shown in Figure 12.2.1. 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 12.2.1 Internet Data Forecast Plots



---

### Example 12.3: 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 criterion=mape;
  forecast cars / model=bestn criterion=mse holdout=5;
  forecast planes / model=addwinters transform=log;
run;
```

---

### Example 12.4: 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 criterion=mape;
  forecast cars / model=bestn criterion=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.

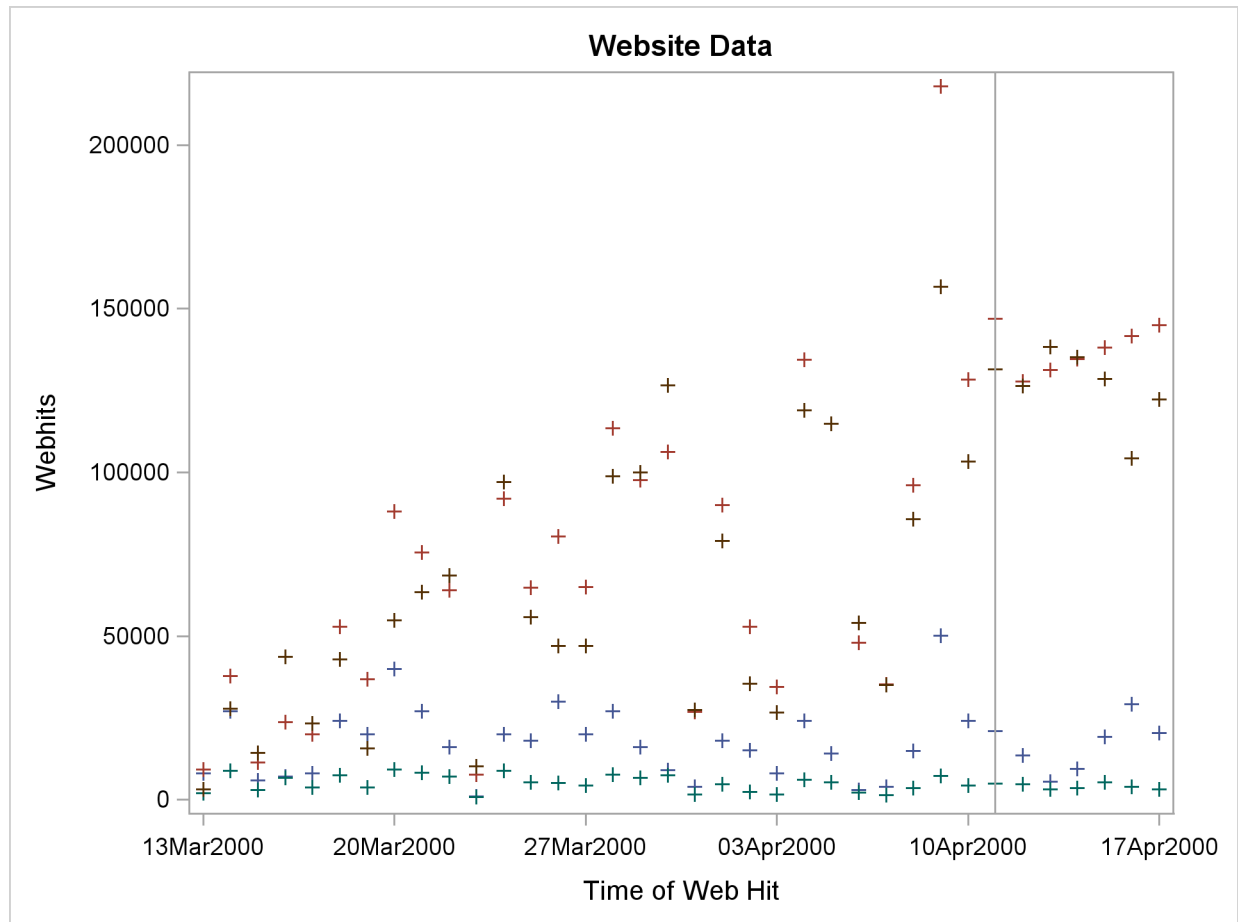
The following GPLOT procedure statements plot the forecasts related to the ENGINES variable:

```

proc sgplot data=enginehits noautolegend;
  scatter x=time y=boats / markerattrs=(symbol=plus);
  scatter x=time y=cars / markerattrs=(symbol=plus);
  scatter x=time y=planes / markerattrs=(symbol=plus);
  scatter x=time y=predicted / markerattrs=(symbol=plus);
  xaxis values=('13MAR2000:00:00:00'dt to '18APR2000:00:00:00'dt by dtweek);
  yaxis label='Webhits';
  refline '11APR2000:00:00:00'dt / axis=x;
run;

```

The GPLOT procedure results are shown in [Figure 12.4.1](#). 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 12.4.1** Internet Data Forecast Plots**Example 12.5: 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;
01JUN1997 0 0 0
01JUL1997 0 1 5
... more lines ...

```

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 12.5.1. The first table records the demand series and predictions. The second table represents forecasts or recommended stocking levels.

**Output 12.5.1** Forecast Tables

Website Data							
The HPF Procedure							
Demands for Variable tires							
Index	Demand	Time	Demand	Demand	Estimate of Mean		
			Intervals	Size	Demand per Period	Std	
			Actual	Actual	Actual	Predict	
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						
Index	Demand	Time	Estimate of Mean Demand per Period			
			95% Confidence Limits	Error	Stock	
1	Sun, 31 Aug 1997		.	.	0.000000	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)

**Output 12.5.1** *continued*

Forecasts for Variable tires					
Obs	Time	Forecasts	Standard 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 12.6: Illustration of ODS Graphics**

This example illustrates the use of ODS graphics.

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

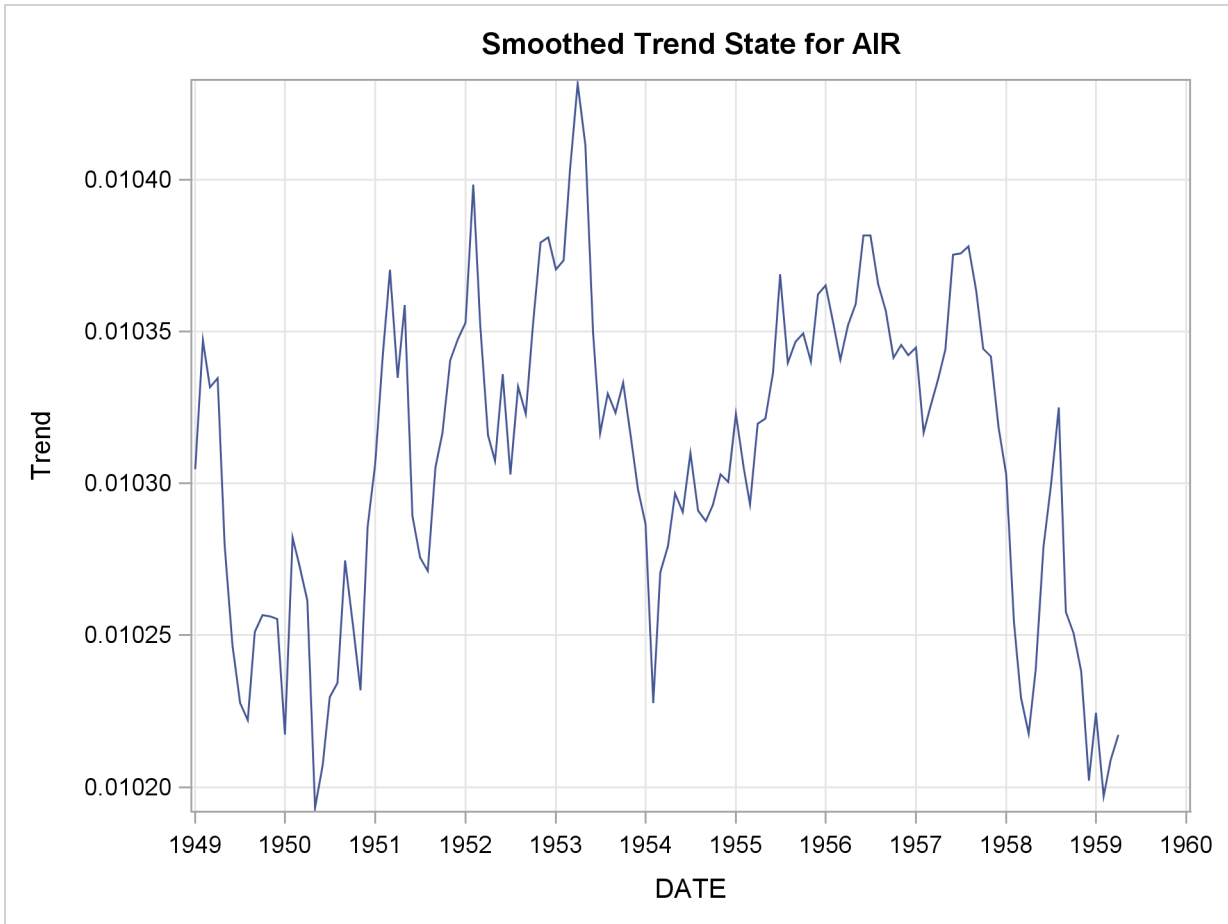
The graphical displays are requested by specifying the ODS GRAPHICS statement and the PLOT= option in the PROC HPF statement. In this case, all plots are requested. [Figure 12.6.1](#) through [Figure 12.6.4](#) show a selection of the plots created.

For specific information about the graphics available in the HPF procedure, see the “[ODS Graphics](#)” on page 364 section.

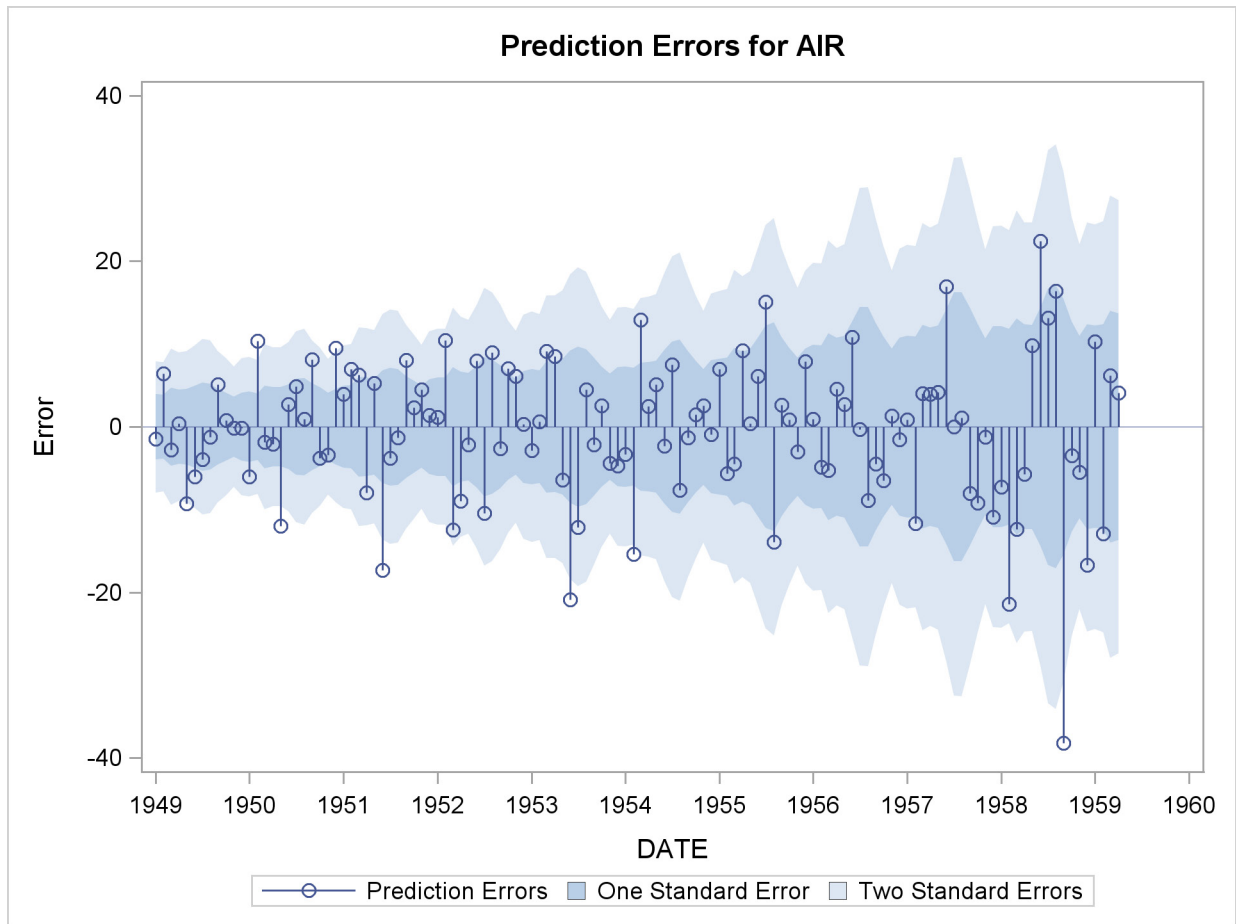
```
proc hpf data=sashelp.air out=_null_
    lead=20
    back=20
    print=all
    plot=all;
    id date interval=month;
    forecast air / model=best transform=auto criterion=mape;
run;
```



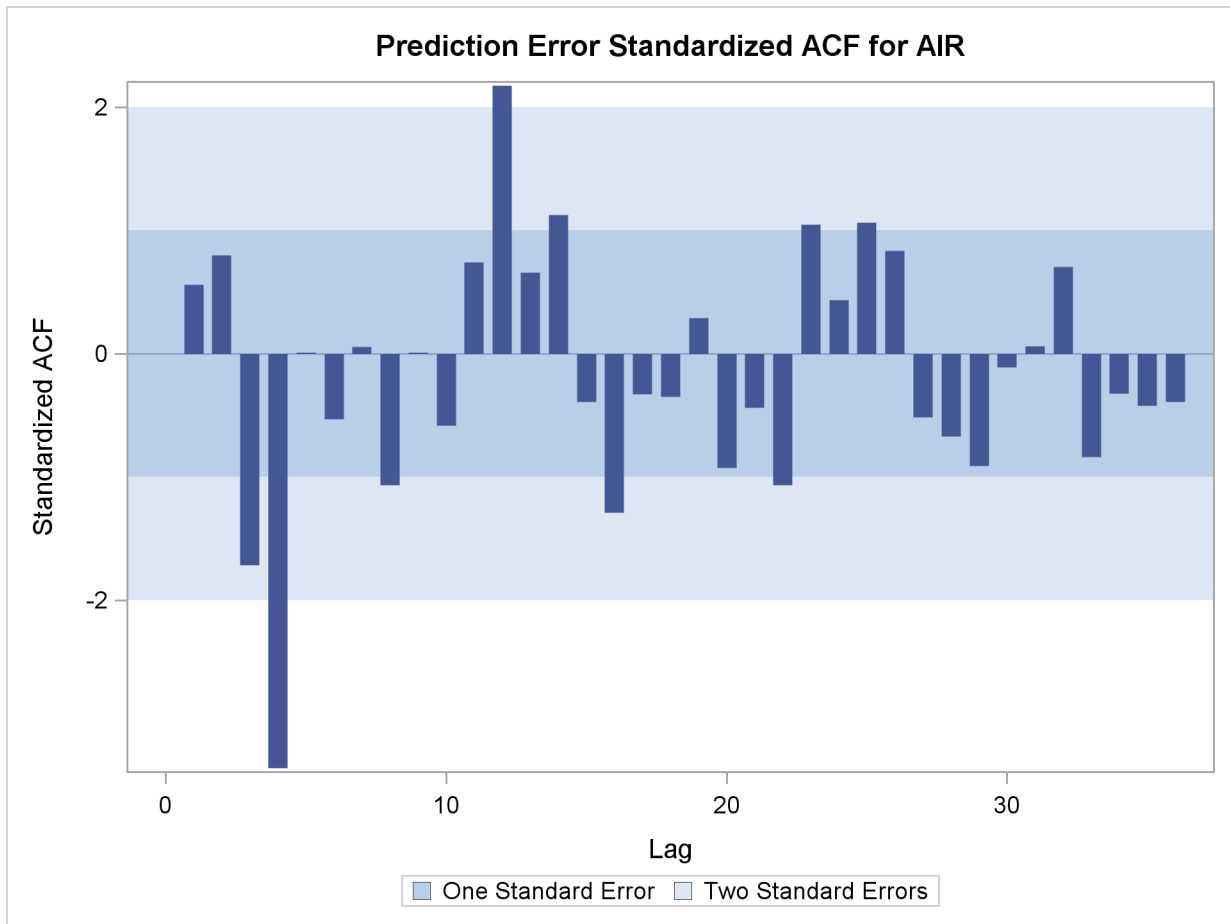
Output 12.6.1 Smoothed Trend Plot

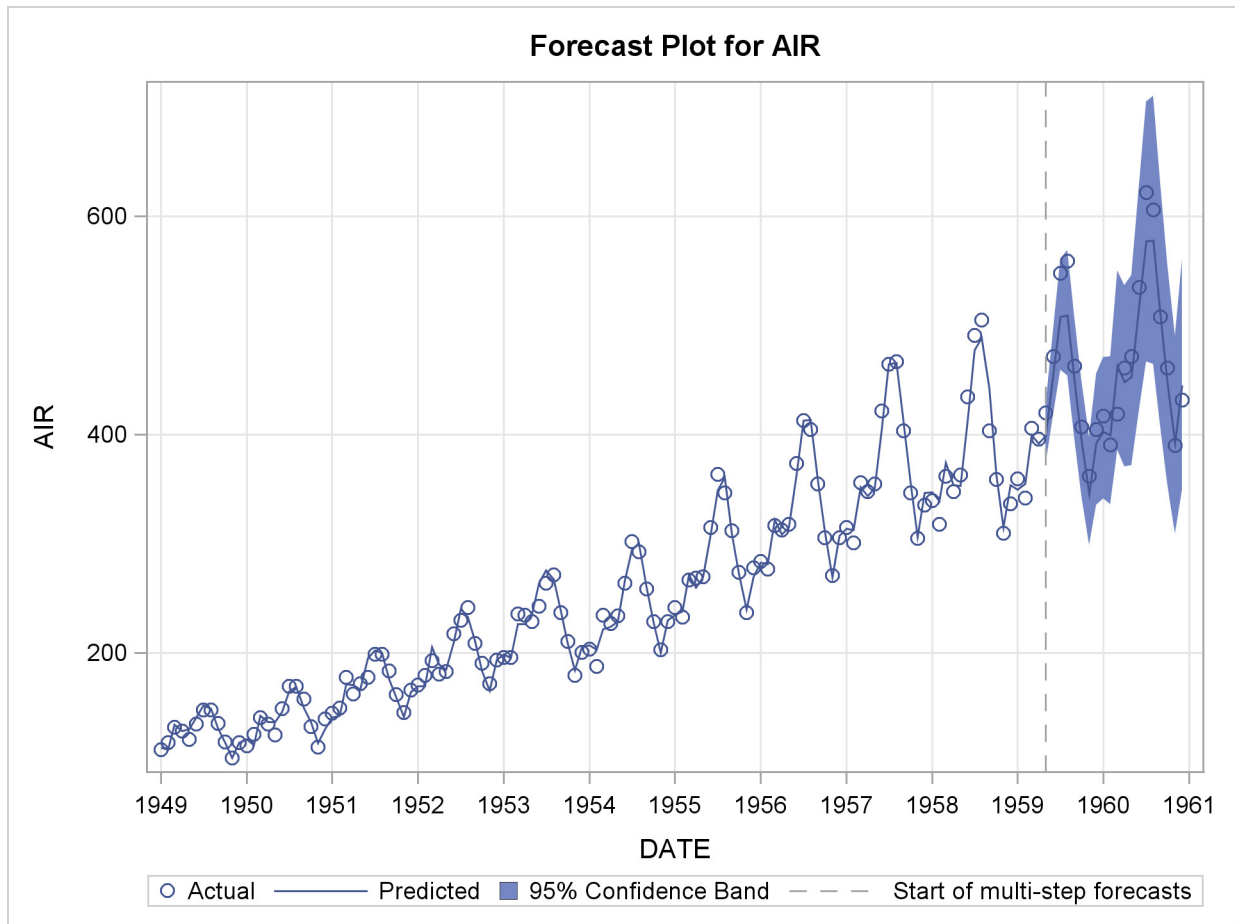


Output 12.6.2 Prediction Error Plot



Output 12.6.3 Prediction Error Standardized ACF Plot



**Output 12.6.4** Forecast Plot

---

## References

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

## **Part III**

# **Forecasting Details**



# Chapter 13

## Forecasting Process Summary

### Contents

---

- Background . . . . . **382**
  - Transactional Data . . . . . 382
  - Time Series Data . . . . . 383
  - Forecasting Models . . . . . 383
  - Forecasts . . . . . 388
  - Forecast Function (Scoring) . . . . . 389
  - Statistics of Fit . . . . . 390
- Automatic Forecasting Process . . . . . **391**
  - Accumulation Step . . . . . 391
  - Interpretation Step . . . . . 391
  - Adjustment Step . . . . . 392
  - Diagnostic Step . . . . . 392
  - Model Selection Step . . . . . 393
  - Parameter Estimation Step . . . . . 393
  - Forecasting Step . . . . . 394
  - Evaluation Step . . . . . 394
  - Performance Step . . . . . 394
  - Forecast Function (Score File) Generation Step . . . . . 395
- Automatic Forecasting Data . . . . . **395**
  - Automatic Forecasting Data Flow . . . . . 395
  - Forecast Scoring Data Flow . . . . . 396
- Automatic Forecasting Information . . . . . **396**
  - Model Specification . . . . . 397
  - Model Selection List . . . . . 399
  - Selected Model Specification . . . . . 400
  - Fitted Model . . . . . 400
  - Forecast Function (Score File) . . . . . 400
  - Automatic Forecasting Information Flow . . . . . 401
  - Forecast Scoring Information Flow . . . . . 402
- Automatic Forecasting Repositories . . . . . **402**
  - Event Repository . . . . . 402
  - Model Specification Repository . . . . . 403
  - Fitted Model Repository . . . . . 405
  - Forecast Results Repository . . . . . 406

Score Repository . . . . .	406
Automatic Forecasting System Flow . . . . .	406
Forecast Scoring System Flow . . . . .	407
Automatic Forecasting Archives . . . . .	<b>409</b>
References . . . . .	<b>409</b>

---

## 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.

$$(\text{Series}) = (\text{Local Level}) + (\text{Error})$$

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.

$$(\text{Series}) = (\text{Local Level}) + (\text{Local Trend}) + (\text{Error})$$

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.

$$(\text{Series}) = (\text{Local Level}) + (\text{Local Season}) + (\text{Error})$$

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.

$$(\text{Series}) = (\text{Local Level}) + (\text{Local Trend}) + (\text{Local Season}) + (\text{Error})$$

$$(\text{Series}) = ((\text{Local Level}) + (\text{Local Trend})) \times (\text{Local Season}) + (\text{Error})$$

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.

$$(\text{Series}) = \text{DisturbanceFilter} (\text{Error})$$

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.

$$(\text{Series}) = \text{TransferFunctionFilter}(\text{Causal Factors}) + \text{DisturbanceFilter}(\text{Error})$$

$$(\text{Series}) = (\text{Local Level}) + (\text{Local Trend}) + (\text{Local Season}) + (\text{Causal Factors}) + (\text{Error})$$

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:

$$(\text{Series}) = \text{Exp} ( (\text{Local Level}) + (\text{Local Trend}) + (\text{Error}) ) \text{ exponential growth model}$$

$$(\text{Series}) = (\text{Local Level}) \times (\text{Local Season}) \times (\text{Error}) \text{ multiplicative error model}$$

Transforming the preceding series permits the use of a linear forecasting model:

$\text{Log}(\text{Series}) = (\text{Local Level}) + (\text{Local Trend}) + (\text{Error})$  log local trend model

$\text{Log}(\text{Series}) = \text{Log}(\text{Local Level}) + \text{Log}(\text{Local Seasonal}) + \text{Log}(\text{Error})$  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 use the forecast score file for scenario (What-If) analysis, stochastic optimization, or goal seeking.

---

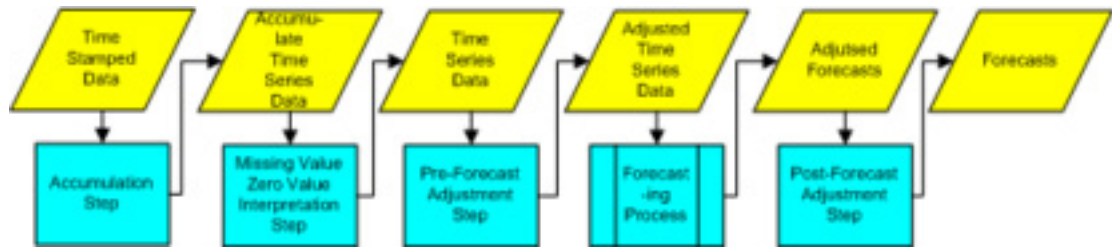
## Automatic Forecasting Data

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

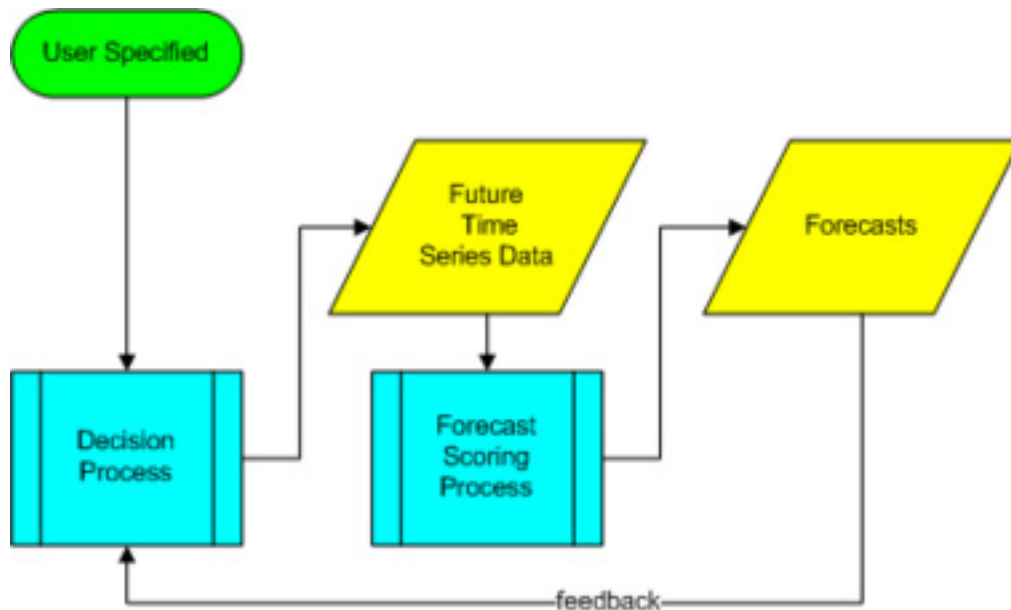
The input and output of the automatic forecasting process is the time-stamped data set and the forecasts, respectively. The following diagram describes the automatic forecasting data flow.

**Figure 13.1** 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 13.2** 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 create 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 non-linear 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 16, “Using 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 10, “[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.

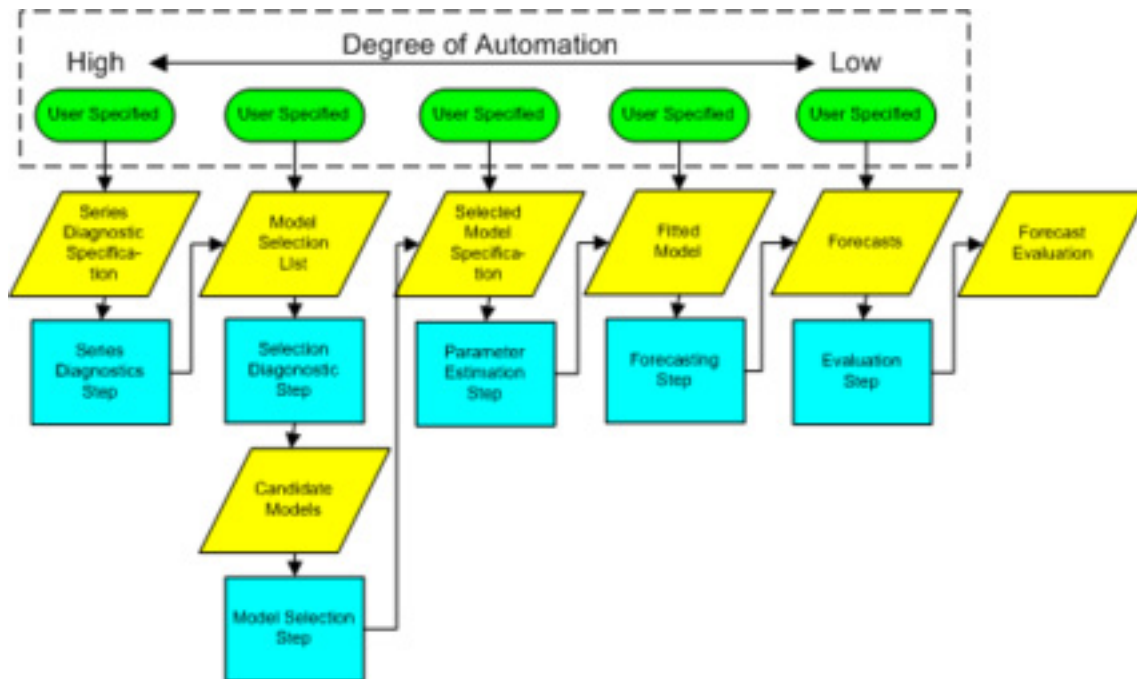
See Chapter 15, “Using Forecasting Model Score Files and DATA Step Functions,” for examples and additional information.

## 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 wish 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 13.3** 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

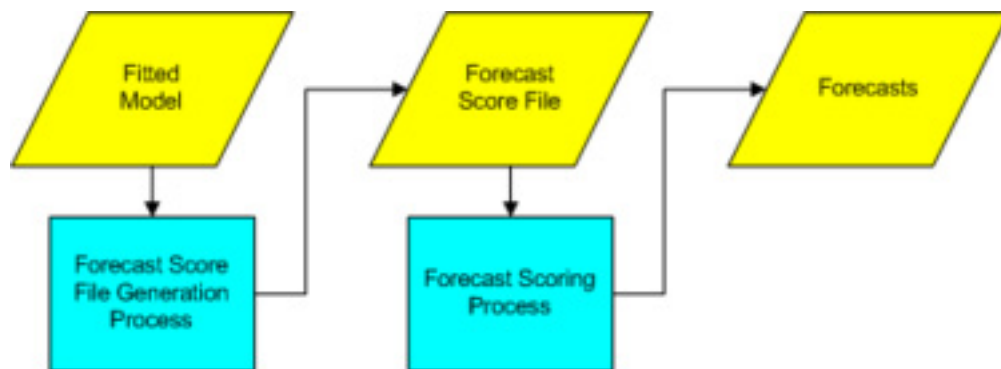
step every week. This staggering of the degree of automation reduces the processing time by allowing the most up-to-date information about the time series data to influence the automatic forecasting process over time.

---

## 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 13.4** 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 6, “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 <i>specifies a time series model to use</i> for forecasting.
Model Selection List File	SAS catalog entry or external file that <i>specifies how to select a model specification</i> to use for a particular time series.

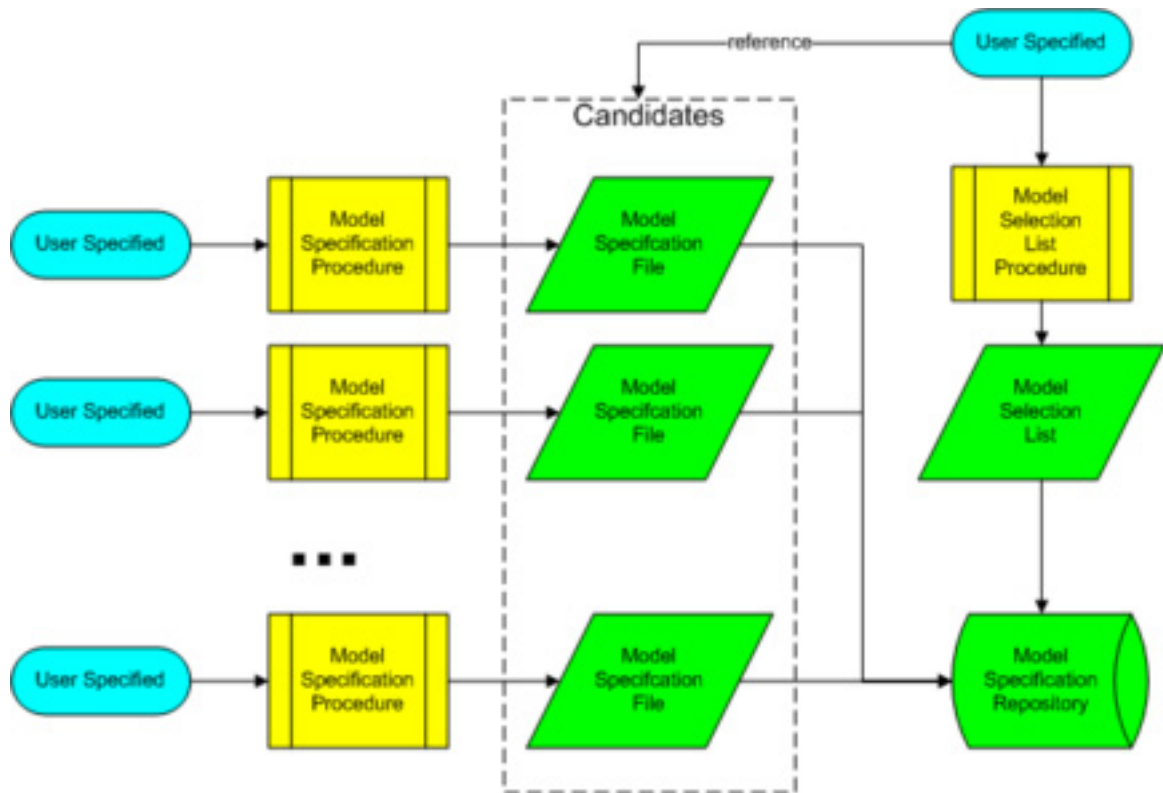
The repository consists of SAS catalogs or external directories (or folders). More than one catalog can be combined using the SAS Libname Engine.

## 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.

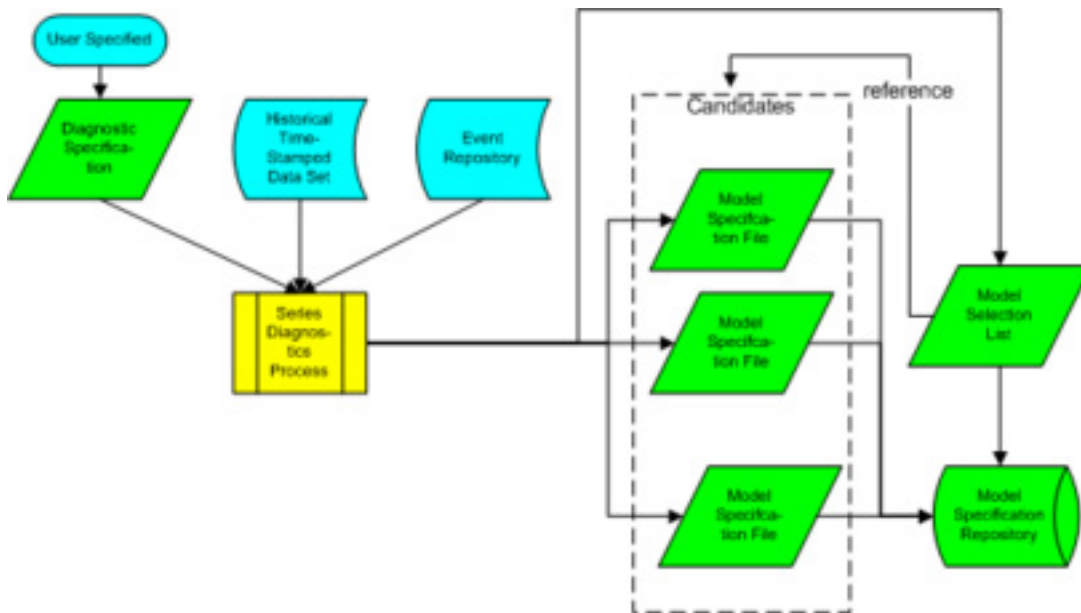
The following diagram illustrates the process of adding user-created model specification files and model selection list files:

Figure 13.5 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 13.6 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)	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

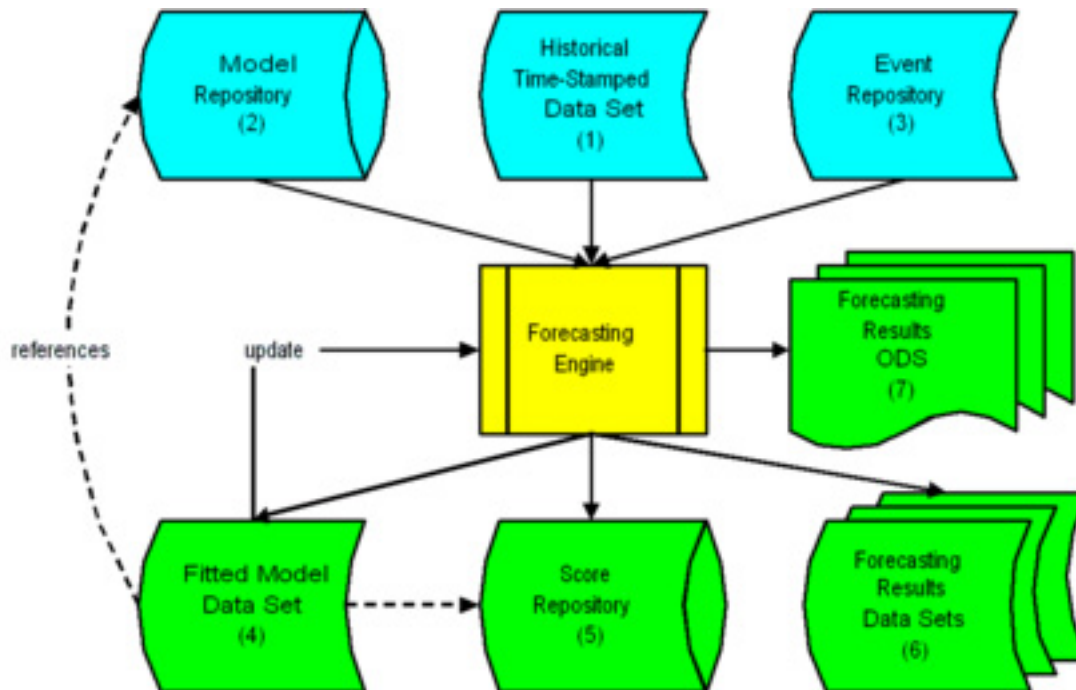
A *score repository* stores information about how to score each time series. Since each time series has a different score, the score repository will often be large because it summarizes information contained in the model specification repository, fitted model repository, as well as the final states (historical 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 13.7](#) shows the system flow for the automatic forecasting technique when there are many time series.



**Figure 13.7** Automatic Forecasting System Flow

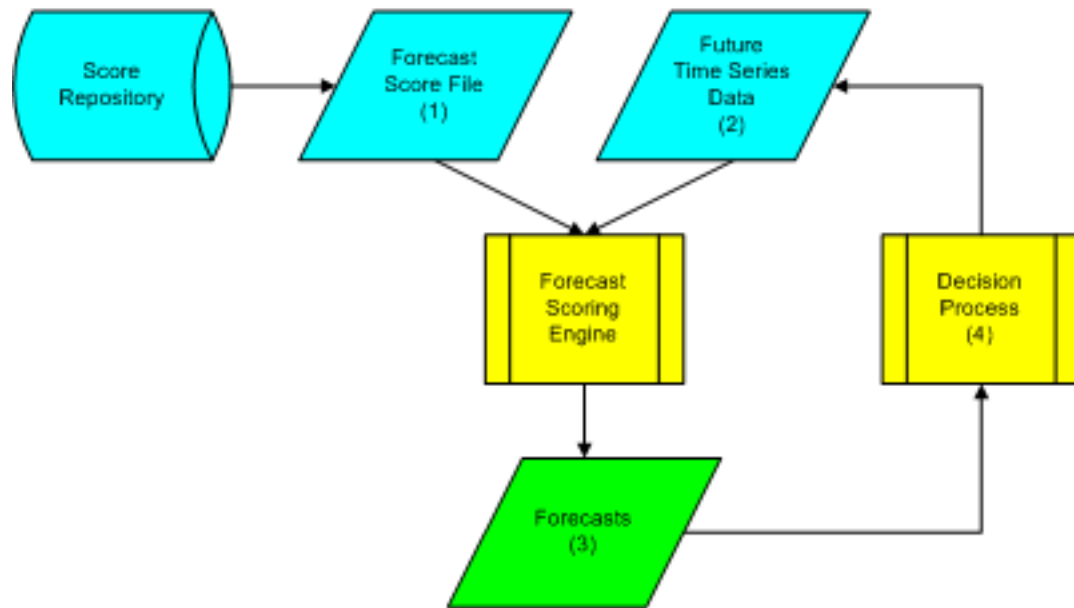
For each historical time series to forecast, the automatic forecasting system works as follows:

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 13.8](#) shows the system flow for each file using the forecast scoring technique.

**Figure 13.8** 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 13.9.

**Figure 13.9** 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 14

## Forecasting Process Details

### Contents

---

Forecasting Process Summary . . . . .	<b>412</b>
Parameter Estimation . . . . .	412
Model Evaluation . . . . .	412
Forecasting . . . . .	412
Smoothing Models . . . . .	<b>413</b>
Smoothing Model Calculations . . . . .	413
Smoothing State and Smoothing Equations . . . . .	413
Smoothing State Initialization . . . . .	414
Missing Values . . . . .	414
Predictions and Prediction Errors . . . . .	414
Smoothing Weights . . . . .	415
Specifying the Smoothing Weights . . . . .	415
Optimizing the Smoothing Weights . . . . .	415
Equations for the Smoothing Models . . . . .	416
ARIMA Models . . . . .	<b>428</b>
ARIMA Model Based Series Decomposition . . . . .	428
UCM Models . . . . .	<b>429</b>
Intermittent Models . . . . .	<b>430</b>
Intermittent Time Series . . . . .	430
Intermittent Series Decomposition and Analysis . . . . .	430
Croston's Method . . . . .	431
Average Demand Method . . . . .	432
Time-Indexed versus Demand-Indexed Holdout Samples . . . . .	433
Automatic Intermittent Demand Model Selection . . . . .	433
External Models . . . . .	<b>434</b>
External Forecasts . . . . .	434
External Forecast Prediction Errors . . . . .	434
External Forecast Prediction Bias . . . . .	435
External Forecast Prediction Standard Errors . . . . .	435
External Forecast Confidence Limits . . . . .	437
Series Transformations . . . . .	<b>437</b>
Predictions for Transformed Models . . . . .	438
Series Diagnostic Tests . . . . .	<b>439</b>
Statistics of Fit . . . . .	<b>440</b>
References . . . . .	<b>443</b>

---

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 6, "The ARIMA Procedure" (*SAS/ETS User's Guide*), and Chapter 28, "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

- $\mu_t$  represents the time-varying mean term.
- $\beta_t$  represents the time-varying slope.
- $s_p(t)$  represents the time-varying seasonal contribution for one of the  $p$  seasons
- $\epsilon_t$  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  $\mu_t$ .
- $T_t$  is a smoothed trend that estimates  $\beta_t$ .
- $S_{t-j}$ ,  $j = 0, \dots, p - 1$ , 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  $j = 0, \dots, p - 1$  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:

- $\alpha$  is a level smoothing weight.
- $\gamma$  is a trend smoothing weight.
- $\delta$  is a seasonal smoothing weight.
- $\phi$  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

$$\text{var}(e_t(k)) = \text{var}(\epsilon_t) \left[ 1 + \sum_{j=1}^{k-1} \alpha^2 \right] = \text{var}(\epsilon_t)(1 + (k-1)\alpha^2)$$

### 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

$$\text{var}(e_t(k)) = \text{var}(\epsilon_t) \left[ 1 + \sum_{j=1}^{k-1} (2\alpha + (j-1)\alpha^2)^2 \right]$$

### 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

$$\text{var}(e_t(k)) = \text{var}(\epsilon_t) \left[ 1 + \sum_{j=1}^{k-1} (\alpha + j\alpha\gamma)^2 \right]$$

## 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

$$\text{var}(e_t(k)) = \text{var}(\epsilon_t) \left[ 1 + \sum_{j=1}^{k-1} (\alpha + \alpha\gamma\phi(\phi^j - 1)/(\phi - 1))^2 \right]$$

### 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)<sub>p</sub> 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

$$\text{var}(e_t(k)) = \text{var}(\epsilon_t) \left[ 1 + \sum_{j=1}^{k-1} \psi_j^2 \right]$$

## Multiplicative Seasonal Smoothing

In order to use the multiplicative version of seasonal smoothing, the time series and all predictions must be strictly positive.

The model equation for the multiplicative version of seasonal 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

$$L_t = L_{t-1} + \alpha e_t / S_{t-p}$$

$$S_t = S_{t-p} + \delta(1 - \alpha)e_t / L_t$$

(Note: For missing values,  $e_t = 0$ .)

The  $k$ -step prediction equation is

$$\hat{Y}_t(k) = L_t S_{t-p+k}$$

The multiplicative version of seasonal smoothing does not have an ARIMA equivalent; however, when the seasonal variation is small, the ARIMA additive-invertible region of the additive version of seasonal described in the preceding section can approximate the stability region of the multiplicative version.

The variance of the prediction errors is estimated as

$$\text{var}(e_t(k)) = \text{var}(\epsilon_t) \left[ \sum_{i=0}^{\infty} \sum_{j=0}^{p-1} (\psi_{j+ip} S_{t+k} / S_{t+k-j})^2 \right]$$

where  $\psi_j$  are as described for the additive version of seasonal method, and  $\psi_j = 0$  for  $j \geq k$ .

## 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)<sub>p</sub> model

$$(1 - B)(1 - B^p)Y_t = \left[ 1 - \sum_{i=1}^{p+1} \theta_i B^i \right] \epsilon_t$$

$$\theta_j = \begin{cases} 1 - \alpha - \alpha\gamma & j = 1 \\ -\alpha\gamma & 2 \leq j \leq p - 1 \\ 1 - \alpha\gamma - \delta(1 - \alpha) & j = p \\ (1 - \alpha)(\delta - 1) & j = p + 1 \end{cases}$$

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

$$\{\max(-p\alpha, 0) < \delta(1 - \alpha) < (2 - \alpha)\}$$

$$\{0 < \alpha\gamma < 2 - \alpha - \delta(1 - \alpha)(1 - \cos(\vartheta))\}$$

where  $\vartheta$  is the smallest nonnegative solution to the equations listed in Archibald (1990).

The variance of the prediction errors is estimated as

$$\text{var}(e_t(k)) = \text{var}(\epsilon_t) \left[ 1 + \sum_{j=1}^{k-1} \psi_j^2 \right]$$

### 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

$$\text{var}(e_t(k)) = \text{var}(\epsilon_t) \left[ \sum_{i=0}^{\infty} \sum_{j=0}^{p-1} (\psi_{j+ip} S_{t+k} / S_{t+k-j})^2 \right]$$

where  $\psi_j$  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 6, “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 $Y_t$
$\mu$	is the constant term
$\phi(B)$	is the autoregressive operator, represented as a polynomial in the back shift operator: $\phi(B) = 1 - \phi_1 B - \dots - \phi_p B^p$
$\theta(B)$	is the moving-average operator, represented as a polynomial in the back shift operator: $\theta(B) = 1 - \theta_1 B - \dots - \theta_q B^q$
$a_t$	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  $\gamma_i$  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 28, “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:

$$\begin{array}{ll}
 \text{(Demand Interval Series)} & q_i = t_i - t_{i-1} \quad \text{for } i = 2 \text{ to } N \\
 \text{(Demand Size Series)} & d_i = y_{t_i} - \text{base} \quad \text{for } i = 1 \text{ to } N \\
 \text{(Average Demand Series)} & a_i = d_i / q_i \quad \text{for } i = 2 \text{ to } N
 \end{array}$$

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:

$$\begin{aligned} \text{(Smoothed demand interval series)} \quad L_i^q &= L_{i-1}^q + \alpha_q(q_{i-1} - L_{i-1}^q) \\ \text{(Smoothed demand size series)} \quad L_i^d &= L_{i-1}^d + \alpha_d(d_{i-1} - L_{i-1}^d) \end{aligned}$$

The demand interval parameter,  $\alpha_q$ , and demand size parameter,  $\alpha_d$ , 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  $\phi_q$  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  $\alpha_q$  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.)

$$\begin{aligned} \text{(Estimated demand per period)} \quad y_i^* &= \hat{d}_i/\hat{q}_i \\ E[y_i^*] &= E[d_i]/E[q_i] = E[z_{i-1}]/E[p_{i-1}] = \mu_d/\mu_q \\ \text{(Variance)} \quad Var(y_i^*) &= (\hat{d}_i/\hat{q}_i)^2(Var(d_i)/\hat{d}_i^2 + Var(q_i)/\hat{q}_i^2) \end{aligned}$$

where  $\mu_d$ ,  $\bar{d}$ , and  $s_d$  are the mean, sample average, and standard deviation of the non-zero demands, and  $\mu_q$ ,  $\bar{q}$ , 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:

$$\begin{aligned} \text{(Recommended stocking level)} \quad \hat{y}_t &= y_i^* && \text{when } t_i = < t < t_{i+1} \\ \text{(Variance)} \quad \text{Var}(\hat{y}_t) &= \text{Var}(y_i^*) && \text{when } t_i = < t < t_{i+1} \end{aligned}$$

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:

$$\begin{aligned} \text{(Accumulated recommended stocks)} \quad s_t &= \sum_{i=0}^t (\hat{y}_t - y_i^*) \\ \text{(Estimate - Demand)} \quad e_{t_i} &= \hat{d}_i q_i - d_i && \text{when time } t_{i+1} \text{ has demand} \end{aligned}$$

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 non-seasonal 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:

$$\text{(Smoothed Average Demand Series)} \quad L_i^a = L_{i-1}^a + \alpha_a (a_{i-1} - L_{i-1}^a)$$

The average demand level smoothing parameter,  $\alpha_a$ , and the starting, intermediate, and final smoothing level states,  $L_i^a$ , are estimated from the data using simple exponential smoothing parameter estimation. For the starting state at  $i = 1$ ,  $L_1^a = \max(a_1, L_0^a)$  where  $L_0^a$  is the final backcast level state. For  $i > 1$ , the one-step-ahead prediction for  $a_i$  is  $\hat{a}_i = L_{i-1}^a$ .

Other nonseasonal exponential smoothing methods can be used in a similar fashion. For linear smoothing,  $L_1^a = \max(a_1 - T_0^a, L_0^a)$  where  $T_0^a$  is the final backcast trend state. For damp-trend smoothing,  $L_1^a = \max(a_1 - \phi_a T_0^a, L_0^a)$  where  $\phi_a$  is the damping parameter and  $T_0^a$  is the final backcast trend state. For double smoothing,  $L_1^a = \max(a_1 - T_0^a / \alpha_a, L_0^a)$  where  $\alpha_a$  is the weight

parameter and  $T_0^a$  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.

$$\begin{aligned} \text{(Estimated demand per period)} \quad & y_i^* = \hat{a}_i + base \\ & E[y_i^*] = E[a_i] \\ \text{(Variance)} \quad & \text{see the exponential smoothing models} \end{aligned}$$

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:

$$\begin{aligned} \text{(Demand Interval Series)} \quad e_i^q &= q_i - \hat{q}_i & \text{for } i = 2 \text{ to } N \\ \text{(Demand Size Series)} \quad e_i^d &= d_i - \hat{d}_i & \text{for } i = 1 \text{ to } N \\ \text{(Average Demand Series)} \quad e_i^a &= a_i - \hat{a}_i & \text{for } i = 2 \text{ to } N \end{aligned}$$

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:

$$\begin{aligned} \text{(Croston's Method)} \quad e_i^c &= (d_i/q_i) - (\hat{d}_i/\hat{q}_i) & \text{for } i = 2 \text{ to } N \\ \text{(Average Demand Method)} \quad e_i^a &= a_i - \hat{a}_i & \text{for } i = 2 \text{ to } N \end{aligned}$$

## 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  $t = 1, \dots, T$  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  $\{\hat{\epsilon}_t\}_{t=1}^T$  to estimate the bias of the forecasts provided by the external source.

$$Bias = \sum_{t=1}^T (y_t - \hat{y}_t) / T = \sum_{t=1}^T \hat{\epsilon}_t / T$$

The prediction mean square error can be used to help judge the significance of this bias.

$$Variance = \sum_{t=1}^T (\hat{\epsilon}_t - Bias)^2 / (T - 1)$$

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 required.)

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)

$$\begin{aligned} \rho(j) &= 1 && \text{for } j = 0 \\ \rho(j) &= 0 && \text{for } j > 0 \end{aligned}$$

This option generates constant prediction error variances.

White Noise Autocorrelation (METHOD=WN)

This option generates white noise prediction error variances.

Error Autocorrelation (Method=ERRORACF)

$$\begin{aligned} \rho(j) &= 1 && \text{for } j = 0 \\ \hat{\rho}(j) &= \frac{1}{T} \sum_{t=j+1}^T \hat{\epsilon}_t \hat{\epsilon}_{t-j} / \hat{\sigma}_\epsilon^2 && \text{for } 0 < j < vT \\ \rho(j) &= 0 && \text{for } j > vT \end{aligned}$$

Series Autocorrelation (METHOD=ACF)

$$\begin{aligned} \rho(j) &= 1 && \text{for } j = 0 \\ \hat{\rho}(j) &= \frac{1}{T} \sum_{t=j+1}^T z_t z_{t-j} / \hat{\sigma}_z^2 && \text{for } 0 < j < vT \\ \rho(j) &= 0 && \text{for } j > vT \end{aligned}$$

$$\text{where } \hat{\sigma}_z^2 = \frac{1}{T} \sum_{t=1}^T (z_t - \bar{z})^2 \text{ and } 0 < v < 1.$$

This option typically generates linear prediction error variances larger than ERRORACF.

Perfect Autocorrelation (METHOD=PERFECT)

$$\rho(j) = 1 \quad \text{for } j \geq 0$$

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  $t = 1, \dots, T$  are the

one-step-ahead prediction standard errors and  $Std(\hat{z}_{T+h}) = \sqrt{\sigma_\epsilon^2 \sum_{j=0}^{h-1} \hat{\rho}(j)^2}$  for  $h = 1, \dots, H$  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,  $\{\hat{z}_t\}_{t=1}^T$ , 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  $\alpha$  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

$$w_t = \ln(y_t)$$

Logistic is the logistic transformation

$$w_t = \ln(cy_t/(1 - cy_t))$$

where the scaling factor  $c$  is

$$c = (1 - e^{-6})10^{-\text{ceil}(\log_{10}(\max(y_t)))}$$

and  $\text{ceil}(x)$  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

$$w_t = \begin{cases} \frac{y_t^\lambda - 1}{\lambda}, & \lambda \neq 0 \\ \ln(y_t), & \lambda = 0 \end{cases}$$

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  $w_t = F(y_t)$  is the transform with inverse-transform  $y_t = F^{-1}(w_t)$ , then

$$\text{median}(\hat{y}_t) = F^{-1}(E[w_t]) = F^{-1}(\hat{w}_t)$$

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  $\sigma_t^2$ , the MMSE predictions for each of the transformations are as follows:



Log is the conditional expectation of inverse-logarithmic transformation.

$$\hat{y}_t = E[e^{w_t}] = \exp(\hat{w}_t + \sigma_t^2/2)$$

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  $c = (1 - 10^{-6})10^{-\text{ceil}(\log_{10}(\max(y_t)))}$ .

Square Root is the conditional expectation of the inverse-square root transformation.

$$\hat{y}_t = E[w_t^2] = \hat{w}_t^2 + \sigma_t^2$$

Box Cox is the conditional expectation of the inverse Box-Cox transformation.

$$\hat{y}_t = \begin{cases} E[(\lambda w_t + 1)^{1/\lambda}], & \lambda \neq 0 \\ E[e^{w_t}] = \exp(\hat{w}_t + \frac{1}{2}\sigma_t^2), & \lambda = 0 \end{cases}$$

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.  $APES = 100 * |y_t - \hat{y}_t| / \sqrt{\sum_{t=1}^n (y_t - \bar{y})^2 / (n - 1)}$  is the absolute error as a percentage of the standard deviation.  $ASE = |y_t - \hat{y}_t| / \frac{1}{n-1} \sum_{t=2}^n |y_t - y_{t-1}|$  is the absolute scaled 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:  $\sum_{t=1}^n (y_t - \bar{y})^2$ , where  $\bar{y}$  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 Error as Percentage of SD.*

The minimum of the absolute error as a percentage of the standard deviation (MINAPES).

*Maximum Absolute Error as Percentage of SD.*

The maximum of the absolute error as a percentage of the standard deviation (MAXAPES).

*Mean Absolute Error as Percentage of SD.*

The mean of the absolute error as a percentage of the standard deviation (MAPES).

*Median Absolute Error as Percentage of SD.*

The median of the absolute error as a percentage of the standard deviation (MdAPES).

*Geometric Mean Error as Percentage of SD.*

The geometric mean of the absolute error as a percentage of the standard deviation (GMAPES).

*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).

*Mean Absolute Scaled Error.*

The mean of the absolute scaled errors (MASE).

*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,  $1 - \left(\frac{n-1}{n-k}\right)(1 - R^2)$ .

*Amemiya's Adjusted R-Square.*

Amemiya's adjusted  $R^2$ ,  $1 - \left(\frac{n+k}{n-k}\right)(1 - R^2)$ .

*Random Walk R-Square.*

The random walk  $R^2$  statistic (Harvey's  $R^2$  statistic using the random walk model for comparison),  $1 - \left(\frac{n-1}{n}\right)SSE/RWSSE$ , 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),  $n \ln(SSE/n) + 2k$ .

*Akaike's Information Criterion, Finite Sample Size Corrected.*

Akaike's information criterion with an empirical correction for small sample sizes (AICC),  $AIC + \frac{2k(k+1)}{n-k-1}$ .

*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,  $\frac{1}{n}SST\left(\frac{n+k}{n-k}\right)(1 - R^2) = \left(\frac{n+k}{n-k}\right)\frac{1}{n}SSE$ .

*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.

Hopwood, 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 15

# Using Forecasting Model Score Files and DATA Step Functions

### Contents

---

Overview . . . . .	447
HPFSCSIG Function . . . . .	447
HPFSCSUB Function . . . . .	449

---

---

## 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-fileref, horizon, ReturnType ) ;

- Scoring-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.

```

data air;
  set sashelp.air;
  controlinput = log(air);
run;

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 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 a;
  sig = hpfscsig( 'score', 3, 'predict' );
  put sig=;
run;

proc print data=a;
run;

```

The hpfcscsig function call produces the result shown in Figure 15.1.

**Figure 15.1** Result of the HPFSCSIG Function Call

Obs	sig
1	HPFSCSUB('XML', 3, 'CONTROLINPUT', ?, ?, ?, 'PREDICT', !, !, !)

In place of XML, you should provide the pre-assigned SAS fileref. You should 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-fileref*, *horizon*, *X1*, *input-1-1*, ..., *input-1-horizon*, *Xj*, *input-j-1*, ..., *input-j-horizon*, *OutputType*, *output-1*, ..., *output-horizon* );

<i>Scoring-Fileref</i>	is a SAS file reference that contains the forecast scoring information.
<i>Horizon</i>	is the forecast horizon or lead. This must be a positive integer value.
<i>X<sub>j</sub></i>	indicates that the next horizon values are the future inputs for controllable variable <i>j</i> .
<i>Input-j-k</i>	is a controllable input value for the <i>j</i> -th variable at horizon <i>k</i> .
<i>OutputType</i>	one of the following: PREDICT, STDERR, LOWER, UPPER. Indicates what will be returned in the output variables.
<i>Output-k</i>	the subroutine output at horizon <i>k</i> .

### 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;

* 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;
```

**Figure 15.2** Output From Example

DATE	AIR	forecast
JAN1961	416.408	416.408
FEB1961	391.715	391.715
MAR1961	419.312	419.312



# Chapter 16

## Using User-Defined Models

### Contents

---

Introduction . . . . .	453
Defining and Using a SAS Language Function or Subroutine . . . . .	454
Defining and Using a C Language External Function . . . . .	457
Input Time Series Keywords . . . . .	461
Returned Forecast Component Keywords . . . . .	461

---

---

### Introduction

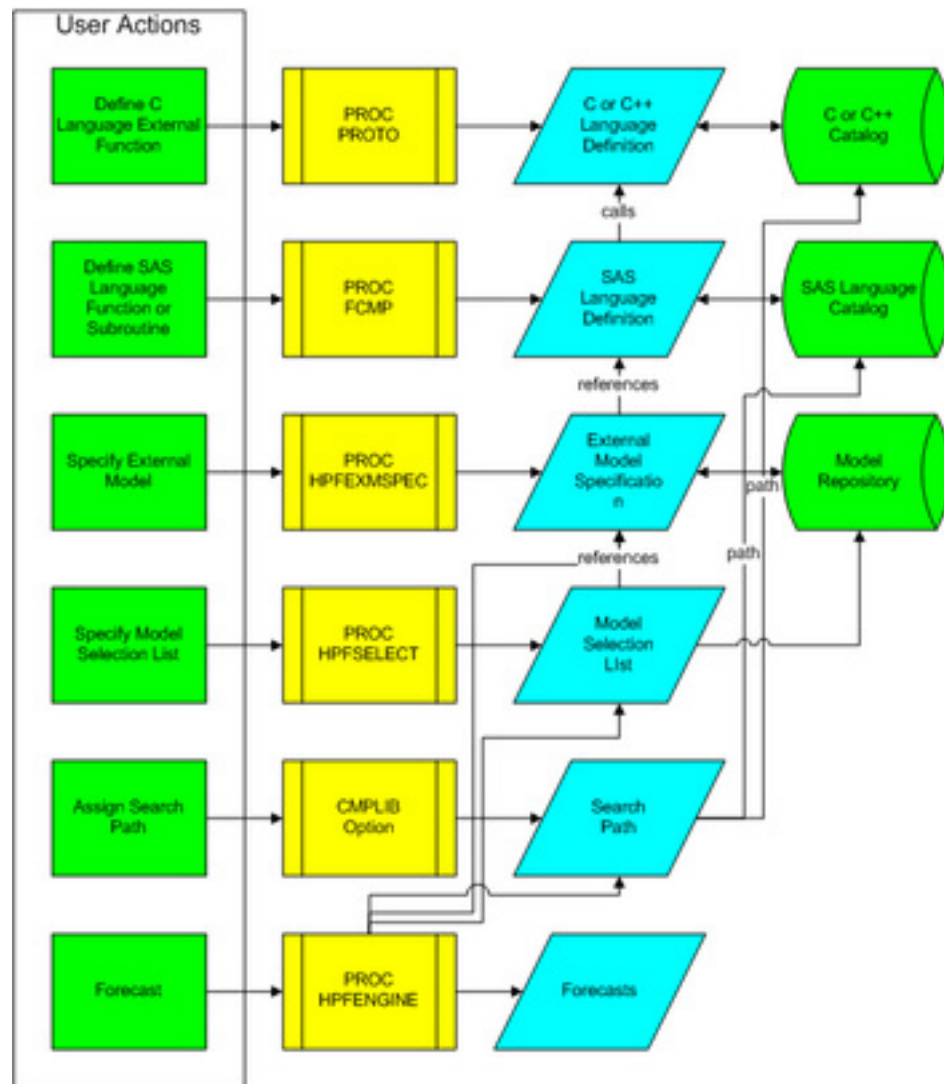
User-defined forecasting models can be used in SAS High-Performance Forecasting. The forecasts produced by these models are considered external forecasts because they are forecasts originating from an external source.

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 16.1 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 WORK.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=work.hpfuser.funcs;
  subroutine lineartrend( actual[*], predict[*], std[*],
                        lower[*], upper[*] );
    outargs 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];
        sume2 = 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 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 WORK.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=work.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 WORK.MYREPOSITORY.

```

proc hpfselect modelrepository=work.myrepository
    selectname=myselect
    selectlabel="My Select List";
    spec lineartrend /
        exmfunc(
            "lineartrend(_actual_ _predict_ _stderr_ _lower_ _upper_)"
        );
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= work.hpfuser;

```

At this point,

1. A SAS language subroutine has been defined, LINEARTREND, and stored in the SAS catalog, WORK.HPFUSER;

2. An external model specification, LINEARTREND, has been stored in the model repository, WORK.MYREPOSITORY;
3. A model selection list, MYSELECT, has been stored in the model repository, WORK.MYREPOSITORY;
4. and the search path for the SAS language functions and subroutines has been set to WORK.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 MODEL-REPOSITORY= WORK.MYREPOSITORY option of the PROC HPFENGINE statement specifies the model repository; the GLOBALSELECTION=MYSELECT options specifies the model selection list; and the

```
proc hpfengine data=sashelp.air
              out=out
              outfor=outfor
              outstat=outstat
              modelrepository=myrepository
              globalselection=myselect;
  id date interval=month;
  forecast air;
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 WORK.CHPFUSER.

```
proc proto package=work.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 ) {

    int      t, n;
    double   value, sumxy, sumx, sumx2, sumy, sumy2;
    double   det, constant, slope, stderr, size, width;
    double   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];
        error = 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 WORK.HPFUSER. The catalog WORK.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=work.hpfuser.funcs
  inlib=work.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 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 WORK.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=work.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 WORK.MYREPOSITORY. 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 hpfselect modelrepository=work.myrepository
              selectname=myselect
              selectlabel="My Select List";
  spec lineartrend /
    exmfunc(
      "lineartrend(_actual_ _predict_ _stderr_ _lower_ _upper_)"
    );
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=( work.hpfuser work.chpfuser );
```

At this point,

1. A C language external function has been defined, LINEARTREND\_C, and stored in the SAS catalog, WORK.CHPFUSER;
2. A SAS language subroutine has been defined, LINEARTREND, which calls the external function, LINEARTREND\_C, and stored in the SAS catalog, WORK.HPFUSER;
3. An external model specification, LINEARTREND, has been stored in the model repository, WORK.MYREPOSITORY;
4. A model selection list, MYSELECT, has been stored in the model repository, WORK.MYREPOSITORY;
5. and the search path for the SAS language functions and subroutines has been set to WORK.HPFUSER and WORK.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=WORK.MYREPOSITORY option of the PROC HPFENGINE statement specifies the model repository; the GLOBALSELECTION=MYSELECT options specifies the model selection list.

```

proc hpfengine data=sashelp.air
              out=out
              outfor=outfor
              outstat=outstat
              modelrepository=myrepository
              globalselection=myselect;
  id date interval=month;
  forecast air;
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 EXMFUNC option of the SPECIFICATION statement in the HPFSELECT procedure. 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

A user-defined forecasting function or subroutine must return the predictions, specified by the keyword `_PREDICT_` in the signature description found in the EXMFUNC option of the SPECIFICATION statement in the HPFSELECT procedure. The prediction standard errors, and the lower and upper confidence limits are optional and are specified by the keywords, `_STDERR_`, `_LOWER_`, and `_UPPER_`, respectively. The HPFENGINE procedure will compute the forecast components that are not returned by the user-defined forecasting function based on the external model specification.





# Chapter 17

## Using External Forecasts

### Contents

---

Introduction . . . . .	463
Specifying and Using an External Model Specification . . . . .	464

---

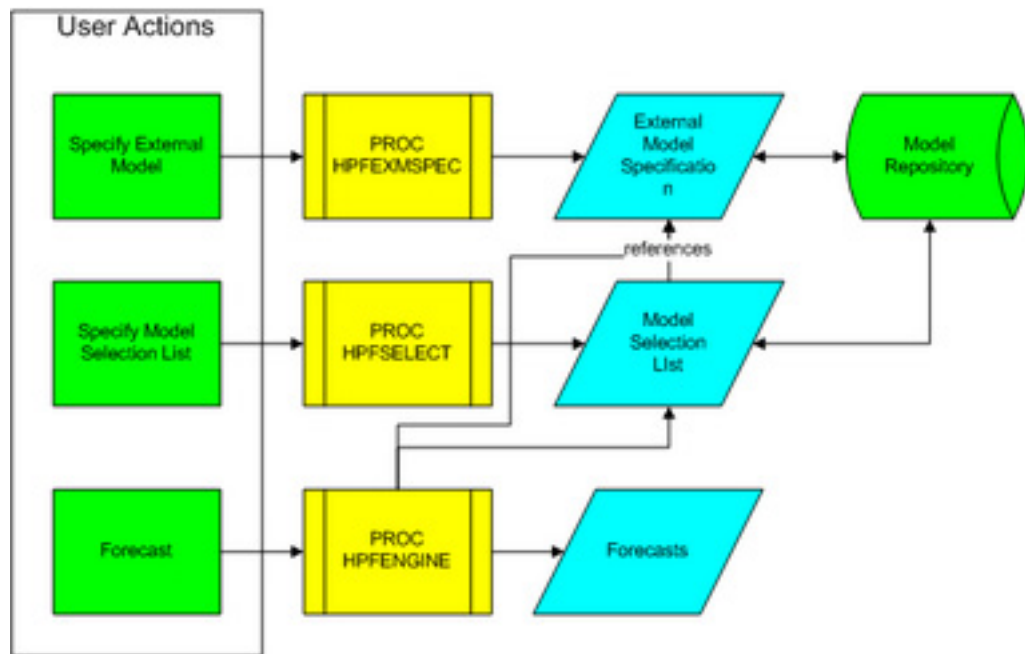
---

### 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 17.1](#) describes an example of the ways external forecasts can be specified and used to forecast.

Figure 17.1 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
    out=engout
    outfor=engforecast
    outstat=outstat
    modelrepository=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

- additive-invertible region
  - smoothing weights, 416
- adjusted R-square
  - statistics of fit, 442
- AIC, *see* Akaike's information criterion
- AICC, *see* Akaike's information criterion, finite sample size corrected
- Akaike's information criterion
  - AIC, 442
  - AICC, 442
  - statistics of fit, 442
- Akaike's information criterion, finite sample size corrected
  - statistics of fit, 442
- Amemiya's prediction criterion
  - statistics of fit, 442
- Amemiya's R-square
  - statistics of fit, 442
- ARIMA models
  - forecasting models, 428
- arma models
  - ARIMA model, 428
- ARMA Order Selection
  - HPFDIAGNOSE procedure, 76
- Base SAS software, 12
- BIC, *see* Schwarz Bayesian information criterion
- boundaries
  - smoothing weights, 416
- Box Cox
  - transformations, 437
- Box Cox transformation, *see* transformations
- Brown smoothing model, *see* double exponential smoothing
- BY groups
  - HPF procedure, 337
  - HPFENGINE procedure, 120
  - HPFEVENTS procedure, 195, 199
  - HPFRECONCILE procedure, 258
- calculations
  - smoothing models, 413
- character functions, 14
- corrected sum of squares
  - statistics of fit, 440
- Croston's, *see* intermittent models
- damped-trend exponential smoothing, 421
  - smoothing models, 421
- DATA step, 12
  - SAS data sets, 12
- diagnostic tests, 439
- Differencing variables in a UCM
  - HPFDIAGNOSE procedure, 82
- double exponential smoothing, 417
  - Brown smoothing model, 417
  - smoothing models, 417
- error sum of squares
  - statistics of fit, 440
- ESTIMATE groups
  - HPFARIMASPEC procedure, 31
- EVENTS
  - HPFDIAGNOSE procedure, 86
- Events in a UCM
  - HPFDIAGNOSE procedure, 86
- Events in an ARIMAX Model
  - HPFDIAGNOSE procedure, 86
- exponential smoothing, *see* smoothing models
- Exponential Smoothing Model
  - HPFDIAGNOSE procedure, 81
- FORECAST groups
  - HPFARIMASPEC procedure, 27
  - HPFUCMSPEC procedure, 314
- forecasting, 412
- forecasting models
  - ARIMA models, 428
  - intermittent models, 430
  - smoothing models, 413
  - UCM models, 429
- Functional Transformation
  - HPFDIAGNOSE procedure, 72
- geometric mean absolute percent error
  - statistics of fit, 441
- geometric mean absolute predictive percent error
  - statistics of fit, 441
- geometric mean absolute symmetric percent error
  - statistics of fit, 441
- geometric mean relative absolute error
  - statistics of fit, 442
- goodness-of-fit statistics, *see* statistics of fit
- help system, 11
- Holt smoothing model, *see* linear exponential smoothing
- Holt-Winters Method, *see* Winters Method

- HPF procedure
  - BY groups, 337
  - ODS graph names, 364
- HPFARIMASPEC procedure
  - ESTIMATE groups, 31
  - FORECAST groups, 27
  - INPUT groups, 29
- HPFDIAGNOSE procedure
  - ARMA Order Selection, 76
  - Differencing variables in a UCM, 82
  - EVENTS, 86
  - Events in a UCM, 86
  - Events in an ARIMAX Model, 86
  - Exponential Smoothing Model, 81
  - Functional Transformation, 72
  - HPFENGINE, 88
  - Intermittent Demand Model, 80
  - Joint Differencing, 75
  - Missing Values, 70
  - OUTEST= Data Set, 90
  - Outliers, 79
  - OUTOUTLIER= Data Set, 92
  - Output Data Sets, 90
  - Season Dummy Test, 75
  - Stationarity Test, 73
  - Transfer Function in a UCM, 82
  - Transfer Function in an ARIMAX Model, 77
  - Unobserved Components Model, 82
- HPFENGINE
  - HPFDIAGNOSE procedure, 88
- HPFENGINE procedure
  - BY groups, 120
  - ODS graph names, 147
- HPFEVENTS procedure
  - BY groups, 195, 199
- HPFRECONCILE procedure
  - BY groups, 258
- HPFUCMSPEC procedure
  - FORECAST groups, 314
  - INPUT groups, 315
  - parameters, 310–314, 316–319
- initializations
  - smoothing models, 414
- INPUT groups
  - HPFARIMASPEC procedure, 29
  - HPFUCMSPEC procedure, 315
- Intermittent Demand Model
  - HPFDIAGNOSE procedure, 80
- intermittent models
  - Croston's Method, 430
  - forecasting models, 430
- Joint Differencing
  - HPFDIAGNOSE procedure, 75
- linear exponential smoothing, 419
  - Holt smoothing model, 419
  - smoothing models, 419
- log
  - transformations, 437
- log test, 439
- log transformation, *see* transformations
- logistic
  - transformations, 437
- maximum absolute percent error
  - statistics of fit, 441
- maximum absolute predictive percent error
  - statistics of fit, 441
- maximum absolute symmetric percent error
  - statistics of fit, 441
- maximum relative absolute error
  - statistics of fit, 442
- mean absolute error
  - statistics of fit, 440
- mean absolute percent error
  - statistics of fit, 441
- mean absolute predictive percent error
  - statistics of fit, 441
- mean absolute scaled error
  - statistics of fit, 442
- mean absolute symmetric percent error
  - statistics of fit, 441
- mean percent error
  - statistics of fit, 443
- mean prediction error
  - statistics of fit, 442
- mean relative absolute error
  - statistics of fit, 442
- mean square error
  - statistics of fit, 440
- median absolute percent error
  - statistics of fit, 441
- median absolute predictive percent error
  - statistics of fit, 441
- median absolute symmetric percent error
  - statistics of fit, 441
- median relative absolute error
  - statistics of fit, 442
- minimum absolute percent error
  - statistics of fit, 441
- minimum absolute predictive percent error
  - statistics of fit, 441
- minimum absolute symmetric percent error
  - statistics of fit, 441
- minimum relative absolute error
  - statistics of fit, 441

- Missing Values
  - HPFDIAGNOSE procedure, 70
- missing values
  - smoothing models, 414
- MMAE, 438
- MMSE, 438
- model evaluation, 412
- multiplicative seasonal smoothing, 424
  - smoothing models, 424
- nonmissing observations
  - statistics of fit, 440
- number of observations
  - statistics of fit, 440
- ODS graph names
  - HPF procedure, 364
  - HPFENGINE procedure, 147
- optimizations
  - smoothing weights, 415
- OUTEST= Data Set
  - HPFDIAGNOSE procedure, 90
- Outliers
  - HPFDIAGNOSE procedure, 79
- OUTOUTLIER= Data Set
  - HPFDIAGNOSE procedure, 92
- Output Data Sets
  - HPFDIAGNOSE procedure, 90
- parameter estimation, 412
- parameters
  - HPFUCMSPEC procedure, 310–314, 316–319
- predictions
  - smoothing models, 414
- R-square statistic
  - statistics of fit, 442
- random walk R-square
  - statistics of fit, 442
- root mean square error
  - statistics of fit, 440
- SAS data sets
  - DATA step, 12
- SBC, *see* Schwarz Bayesian information criterion
- Schwarz Bayesian information criterion
  - BIC, 442
  - SBC, 442
  - statistics of fit, 442
- Season Dummy Test
  - HPFDIAGNOSE procedure, 75
- seasonal exponential smoothing, 422
  - smoothing models, 422
- seasonality test, 439
- series diagnostics, 439
- simple exponential smoothing, 416
  - smoothing models, 416
- smoothing equations, 413
  - smoothing models, 413
- smoothing models
  - calculations, 413
  - damped-trend exponential smoothing, 421
  - double exponential smoothing, 417
  - exponential smoothing, 413
  - forecasting models, 413
  - initializations, 414
  - linear exponential smoothing, 419
  - missing values, 414
  - multiplicative seasonal smoothing, 424
  - predictions, 414
  - seasonal exponential smoothing, 422
  - simple exponential smoothing, 416
  - smoothing equations, 413
  - smoothing state, 413
  - smoothing weights, 415
  - standard errors, 416
  - underlying model, 413
  - Winters Method, 425, 426
- smoothing state, 413
  - smoothing models, 413
- smoothing weights, 415
  - additive-invertible region, 416
  - boundaries, 416
  - optimizations, 415
  - smoothing models, 415
  - specifications, 415
  - weights, 415
- specifications
  - smoothing weights, 415
- square root
  - transformations, 437
- square root transformation, *see* transformations
- standard errors
  - smoothing models, 416
- Stationarity Test
  - HPFDIAGNOSE procedure, 73
- statistics of fit, 440
  - adjusted R-square, 442
  - Akaike's information criterion, 442
  - Akaike's information criterion, finite sample size corrected, 442
  - Amemiya's prediction criterion, 442
  - Amemiya's R-square, 442
  - corrected sum of squares, 440
  - error sum of squares, 440
  - geometric mean absolute percent error, 441
  - geometric mean absolute predictive percent error, 441

- geometric mean absolute symmetric percent error, 441
- geometric mean relative absolute error, 442
- goodness-of-fit statistics, 440
- maximum absolute percent error, 441
- maximum absolute predictive percent error, 441
- maximum absolute symmetric percent error, 441
- maximum relative absolute error, 442
- mean absolute error, 440
- mean absolute percent error, 441
- mean absolute predictive percent error, 441
- mean absolute scaled error, 442
- mean absolute symmetric percent error, 441
- mean percent error, 443
- mean prediction error, 442
- mean relative absolute error, 442
- mean square error, 440
- median absolute percent error, 441
- median absolute predictive percent error, 441
- median absolute symmetric percent error, 441
- median relative absolute error, 442
- minimum absolute percent error, 441
- minimum absolute predictive percent error, 441
- minimum absolute symmetric percent error, 441
- minimum relative absolute error, 441
- nonmissing observations, 440
- number of observations, 440
- R-square statistic, 442
- random walk R-square, 442
- root mean square error, 440
- Schwarz Bayesian information criterion, 442
- uncorrected sum of squares, 440
- forecasting models, 429
- uncorrected sum of squares
  - statistics of fit, 440
- underlying model
  - smoothing models, 413
- Unobserved Components Model
  - HPFDIAGNOSE procedure, 82
- weights, *see* smoothing weights
- Winters Method, 425, 426
  - Holt-Winters Method, 425
  - smoothing models, 425, 426
- Time Series Reconciliation, *see also* HPFRECONCILE procedure
- Transfer Function in a UCM
  - HPFDIAGNOSE procedure, 82
- Transfer Function in an ARIMAX Model
  - HPFDIAGNOSE procedure, 77
- transformations
  - Box Cox, 437
  - Box Cox transformation, 437
  - log, 437
  - log transformation, 437
  - logistic, 437
  - square root, 437
  - square root transformation, 437
- UCM models



# Syntax Index

- ABEL= option
  - PROC HPFIDMSPEC statement, 239
- ACCUMULATE= option
  - ADJUST statement, 58
  - ADJUST statement (ENG), 120
  - CONTROL statement (ENG), 121
  - FORECAST statement, 62
  - FORECAST statement (ENG), 122
  - FORECAST statement (HPF), 338
  - ID statement, 63
  - ID statement (ENG), 124
  - ID statement (HPF), 343
  - INPUT statement, 66
  - INPUT statement (ENG), 127
  - STOCHASTIC statement (ENG), 129
- ADJUST statement
  - HPFDIAGNOSE procedure, 57
  - HPFENGINE procedure, 119
- AFTER= option
  - EVENTDEF statement (HPFEVENTS), 197
- AGGBY statement
  - HPFRECONCILE procedure, 257
- AGGDATA Statement
  - HPFRECONCILE procedure, 257
- AGGDATA=option
  - PROC HPFRECONCILE statement, 254
- AGGREGATE= option
  - PROC HPFRECONCILE statement, 255
- ALIGN= option
  - ID statement, 63
  - ID statement (ENG), 125
  - ID statement (HPF), 344
  - ID statement (HPFEVENTS), 202
  - ID statement (HPFRECONCILE), 260
- ALPHA= option
  - FORECAST statement (HPF), 338
  - FORECASTOPTIONS statement (HPFSELECT), 286
  - PROC HPFDIAGNOSE statement, 50
  - PROC HPFRECONCILE statement, 255
- AR= option
  - FORECAST statement (HPFARIMASPEC), 28
  - INPUT statement (HPFARIMASPEC), 30
- ARIMAX statement
  - HPFDIAGNOSE procedure, 58
- AUTOREG statement
  - HPFUCMSPEC procedure, 309
- AVERAGE= option
  - IDM statement (HPF), 347
  - IDM statement (HPFIDMSPEC), 240
- BACK= option
  - PROC HPF statement, 334
  - PROC HPFDIAGNOSE statement, 51
  - PROC HPFENGINE statement, 110
- BASE= option
  - IDM statement, 65
  - IDM statement (HPF), 347
  - IDM statement (HPFIDMSPEC), 240
- BASENAME= option
  - PROC HPFDIAGNOSE statement, 51
- BEFORE= option
  - EVENTDEF statement (HPFEVENTS), 197
- BLOCKSEASON statement
  - HPFUCMSPEC procedure, 310
- BLOCKSIZE= option
  - BLOCKSEASON statement (HPFUCMSPEC), 311
- Bottom-up Reconciliation
  - HPFRECONCILE procedure, 265
- BOUNDS= option
  - IDM statement (HPF), 349
  - IDM statement (HPFIDMSPEC), 241
- BY statement
  - HPF procedure, 337
  - HPFDIAGNOSE procedure, 61
  - HPFENGINE procedure, 120
  - HPFEVENTS procedure, 195
  - HPFRECONCILE procedure, 258
- CHOOSE= specification
  - SELECT statement (HPFSELECT), 287
- CLMETHOD= option
  - PROC HPFRECONCILE statement, 255
- COMPONENT= option
  - UCM statement, 68
- CONDENSE option
  - EVENTDATA statement, 196
- Confidence Limits
  - Bottom-up Reconciliation (HPFRECONCILE), 266
  - Top-Down Reconciliation (HPFRECONCILE), 264
- CONSTRAINT= option
  - PROC HPFRECONCILE statement, 254
- CONTROL statement

- HPFENGINE procedure, 120
- CONVERGE= option
  - ESTIMATE statement (HPFARIMASPEC), 31
- CRITERION= option
  - ARIMAX statement, 58
  - ESM statement (HPFESMSPEC), 178
  - FORECAST statement (HPF), 338
  - IDM statement (HPF), 350
  - IDM statement (HPFIDMSPEC), 241
  - PROC HPFDIAGNOSE statement, 51
  - SELECT statement (HPFSELECT), 287
- CYCLE statement
  - HPFUCMSPEC procedure, 311
- DAMPPARM= option
  - ESM statement (HPFESMSPEC), 179
  - IDM statement (HPF), 349
  - IDM statement (HPFIDMSPEC), 242
- DAMPREST= option
  - ESM statement (HPFESMSPEC), 179
  - IDM statement (HPF), 349
  - IDM statement (HPFIDMSPEC), 242
- DATA= option
  - PROC HPF statement, 334
  - PROC HPFDIAGNOSE statement, 52
  - PROC HPFENGINE statement, 110
  - PROC HPFEVENTS statement, 195
- DELAY= option
  - INPUT statement (HPFARIMASPEC), 30
  - INPUT statement (HPFUCMSPEC), 315
- DELAYEVENT= option
  - PROC HPFDIAGNOSE statement, 52
- DELAYINPUT= option
  - PROC HPFDIAGNOSE statement, 52
- DELETE statement
  - HPFSELECT procedure, 285
- DELTA= option
  - ESTIMATE statement (HPFARIMASPEC), 31
- DEN= option
  - ARIMAX statement, 58
  - INPUT statement (HPFARIMASPEC), 30
- DEPLAG statement
  - HPFUCMSPEC procedure, 313
- DIAGNOSE statement
  - HPFSELECT procedure, 285
- DIF= option
  - FORECAST statement (HPFARIMASPEC), 27
  - INPUT statement (HPFARIMASPEC), 30
  - INPUT statement (HPFUCMSPEC), 315
- DIFF= option
  - TREND statement, 67
- DIRECTION= option
  - PROC HPFRECONCILE statement, 256
- DISAGGDATA Statement
  - HPFRECONCILE procedure, 258
- DISAGGDATA=option
  - PROC HPFRECONCILE statement, 254
- DISAGGREGATION= option
  - PROC HPFRECONCILE statement, 256
- DURATION= option
  - EVENTDEF statement (HPFEVENTS), 197
- END= option
  - ID statement, 63
  - ID statement (ENG), 125
  - ID statement (HPF), 344
  - ID statement (HPFEVENTS), 202
  - ID statement (HPFRECONCILE), 259
- ENTRYPCT= option
  - PROC HPFDIAGNOSE statement, 52
- ERRORCONTROL= option
  - PROC HPFDIAGNOSE statement, 53
  - PROC HPFENGINE statement, 118
- ERRORTRACE= option
  - PROC HPFRECONCILE statement, 255
- ESM statement
  - HPFDIAGNOSE procedure, 61
  - HPFESMSPEC procedure, 178
- ESTIMATE statement
  - HPFARIMASPEC procedure, 31
- ESTMETHOD= option
  - ARIMAX statement, 59
- EVENT statement
  - HPFDIAGNOSE procedure, 61
- EVENT= option
  - PROC HPFENGINE statement, 110
- EVENTBY= option
  - PROC HPFDIAGNOSE statement, 53
- EVENTCOMB statement
  - HPFEVENTS procedure, 195
- EVENTDATA statement
  - HPFEVENTS procedure, 196
- EVENTDEF statement
  - HPFEVENTS procedure, 196
- EVENTDUMMY statement
  - HPFEVENTS procedure, 199
- EVENTGROUP statement
  - HPFEVENTS procedure, 199
- EVENTKEY statement
  - HPFEVENTS procedure, 201
- EVENTMAP option
  - SPECIFICATIONS statement (HPFSELECT), 289
- EXM statement
  - HPFEXMSPEC procedure, 233

- EXMFUNC option
  - SPECIFICATIONS statement (HPFSELECT), 290
- EXMMAP option
  - SPECIFICATIONS statement (HPFSELECT), 289
- EXTEND= option
  - CONTROL statement (ENG), 121
- EXTERNAL statement
  - HPFENGINE procedure, 122
- FORCECONSTRAINT option
  - PROC HPFRECONCILE statement, 254
- FORECAST statement
  - HPF procedure, 338
  - HPFARIMASPEC procedure, 27
  - HPFDIAGNOSE procedure, 62
  - HPFENGINE procedure, 122
  - HPFUCMSPEC procedure, 314
- FORECASTOPTIONS statement
  - HPFSELECT procedure, 286
- FORMAT= option
  - ID statement (ENG), 125
  - ID statement (HPF), 344
  - ID statement (HPFEVENTS), 202
- GLOBALSELECTION= option
  - PROC HPFENGINE statement, 110
- HOLDOUT= option
  - FORECAST statement (HPF), 339
  - PROC HPFDIAGNOSE statement, 54
  - SELECT statement (HPFSELECT), 288
- HOLDOUTPCT= option
  - FORECAST statement (HPF), 340
  - PROC HPFDIAGNOSE statement, 54
  - SELECT statement (HPFSELECT), 288
- HORIZONSTART= option
  - ID statement (ENG), 125
- HPF, 328
- HPF procedure, 332
  - syntax, 332
- HPFDIAGNOSE procedure, PROC
  - HPFDIAGNOSE statement
  - OUTPROCINFO= option, 55
- HPFENGINE procedure, 107
  - syntax, 107
- HPFESMSPEC procedure, 176
  - syntax, 176
- HPFEVENTS procedure, 193
  - syntax, 193
- HPFEXMSPEC, 231
- HPFEXMSPEC procedure, 232
  - syntax, 232
- HPFIDMSPEC, 175, 237
  - HPFIDMSPEC procedure, 238
    - syntax, 238
- HPFRECONCILE procedure
  - syntax, 251
- HPFRECONCILE procedure, PROC
  - HPFRECONCILE statement
    - AGGDATA= option, 254
    - AGGREGATE= option, 255
    - ALPHA= option, 255
    - CLMETHOD= option, 255
    - CONSTRAINT= option, 254
    - DIRECTION= option, 256
    - DISAGGDATA= option, 254
    - DISAGGREGATION= option, 256
    - ERRORTRACE= option, 255
    - FORCECONSTRAINT option, 254
    - OUTFOR= option, 254
    - OUTINFEASIBLE= option, 254
    - OUTPROCINFO= option, 255
    - PREDICTONLY option, 256
    - RECDIFF option, 255
    - SIGN= option, 256
    - STDDIFBD= option, 257
    - STDMETHOD= option, 256
    - WEIGHTED option, 257
- HPFSELECT, 281
- HPFSELECT procedure, 283
  - syntax, 283
- ID statement
  - HPF procedure, 343
  - HPFDIAGNOSE procedure, 63
  - HPFENGINE procedure, 123
  - HPFEVENTS procedure, 202
  - HPFRECONCILE procedure, 259
- IDENTIFYORDER= option
  - ARIMAX statement, 60
- IDM statement
  - HPF procedure, 346
  - HPFDIAGNOSE procedure, 64
  - HPFIDMSPEC procedure, 240
- IDMBASE= option
  - DIAGNOSE statement (HPFSELECT), 285
- IN= option
  - EVENTDATA statement, 196
- INEST= option
  - PROC HPFENGINE statement, 110
- INEVENT= option
  - PROC HPFDIAGNOSE statement, 54
- INPUT statement
  - HPFARIMASPEC procedure, 29
  - HPFDIAGNOSE procedure, 65
  - HPFENGINE procedure, 127
  - HPFUCMSPEC procedure, 315

- INPUTMAP option
  - SPECIFICATIONS statement (HPFSELECT), 291
- INSELECTNAME= option
  - PROC HPFDIAGNOSE statement, 54
  - PROC HPFSELECT statement, 284
- INTERMITTENT= option
  - DIAGNOSE statement (HPFSELECT), 286
  - FORECAST statement (HPF), 340
  - IDM statement, 65
- INTERVAL= option
  - ID statement, 63
  - ID statement (HPF), 344
  - ID statement (HPFENGINE), 125
  - ID statement (HPFEVENTS), 202
  - ID statement (HPFRECONCILE), 259
  - IDM statement (HPF), 348
  - IDM statement (HPFIDMSPEC), 240
- IRREGULAR option
  - ID statement (HPFRECONCILE), 259
- IRREGULAR statement
  - HPFUCMSPEC procedure, 316
- LABEL= option
  - EVENTCOMB statement (HPFEVENTS), 195, 200
  - EVENTDEF statement (HPFEVENTS), 197
  - PROC HPFESMSPEC statement, 177
  - PROC HPFEXMSPEC statement, 233
  - PROC HPFSELECT statement, 284
  - SPECIFICATIONS statement (HPFSELECT), 291
- LAGS= option
  - DEPLAG statement (HPFUCMSPEC), 313
- LEAD= option
  - PROC HPF statement, 334
  - PROC HPFENGINE statement, 110
  - PROC HPFEVENTS statement, 195
- LENGTH= option
  - SEASON statement (HPFUCMSPEC), 318
- LEVEL statement
  - HPFUCMSPEC procedure, 316
- LEVELPARM= option
  - ESM statement (HPFESMSPEC), 179
  - IDM statement (HPF), 349
  - IDM statement (HPFIDMSPEC), 242
- LEVELREST= option
  - ESM statement (HPFESMSPEC), 179
  - IDM statement (HPF), 349
  - IDM statement (HPFIDMSPEC), 243
- MA= option
  - FORECAST statement (HPFARIMASPEC), 28
- MAXERROR= option
  - PROC HPF statement, 334
  - PROC HPFEVENTS statement, 195
- MAXIT= option
  - ESTIMATE statement (HPFARIMASPEC), 31
- MAXITER= option
  - ESTIMATE statement (HPFARIMASPEC), 31
- MEDIAN option
  - ESM statement (HPFESMSPEC), 180
  - EXM statement (HPFEXMSPEC), 234
  - FORECAST statement (HPF), 340
  - IDM statement (HPF), 349
  - IDM statement (HPFIDMSPEC), 243
- METHOD= option
  - ARIMAX statement, 59
  - ESM statement, 61
  - ESM statement (HPFESMSPEC), 180
  - EXM statement (HPFEXMSPEC), 234
  - IDM statement (HPF), 349
  - IDM statement (HPFIDMSPEC), 243
- METHOD=CLS option
  - ESTIMATE statement (HPFARIMASPEC), 31
- METHOD=ML option
  - ESTIMATE statement (HPFARIMASPEC), 31
- METHOD=ULS option
  - ESTIMATE statement (HPFARIMASPEC), 31
- MINOBS=(SEASON= ) option
  - PROC HPFDIAGNOSE statement, 54
- MINOBS=(TREND= ) option
  - PROC HPFDIAGNOSE statement, 54
- Missing Values
  - Bottom-up Reconciliation (HPFRECONCILE), 265
  - Top-Down Reconciliation (HPFRECONCILE), 263
- MODEL= option
  - FORECAST statement (HPF), 340
- MODELREP= option
  - PROC HPFESMSPEC statement, 177
- MODELREPOSITORY= option
  - PROC HPFESMSPEC statement, 177
- MU= option
  - FORECAST statement (HPFARIMASPEC), 28
- NAME= option
  - PROC HPFESMSPEC statement, 177
  - PROC HPFEXMSPEC statement, 233
  - PROC HPFIDMSPEC statement, 239

- PROC HPFSELECT statement, 284
- NBACKCAST= option
  - FORECAST statement (HPF), 341
- NBLOCKS= option
  - BLOCKSEASON statement (HPFUCMSPEC), 311
- NLAGPCT= option
  - EXM statement (HPFEXMSPEC), 234
- NOCONSTANT option
  - FORECAST statement (HPFARIMASPEC), 28
- NODIAGNOSE option
  - PROC HPFDIAGNOSE statement, 54
- NOEST option
  - AUTOREG statement (HPFUCMSPEC), 310
  - BLOCKSEASON statement (HPFUCMSPEC), 311
  - CYCLE statement (HPFUCMSPEC), 312
  - DEPLAG statement (HPFUCMSPEC), 314
  - ESM statement (HPFESMSPEC), 180
  - ESTIMATE statement (HPFARIMASPEC), 31
  - EXM statement (HPFEXMSPEC), 234
  - IDM statement (HPF), 349
  - IDM statement (HPFIDMSPEC), 243
  - IRREGULAR statement (HPFUCMSPEC), 316
  - LEVEL statement (HPFUCMSPEC), 317
  - SEASON statement (HPFUCMSPEC), 318
  - SLOPE statement (HPFUCMSPEC), 319
- NOINESTOPTS option
  - PROC HPFDIAGNOSE statement, 54
- NOINT option
  - FORECAST statement (HPFARIMASPEC), 28
- NOISEVAR= option
  - FORECAST statement (HPFARIMASPEC), 28
- NOLS option
  - ESTIMATE statement (HPFARIMASPEC), 31
- NOOUTALL option
  - PROC HPF statement, 334
- NOREPLACE option
  - FORECAST statement (ENG), 123
- NOSTABLE option
  - ESM statement (HPFESMSPEC), 180
  - ESTIMATE statement (HPFARIMASPEC), 31
  - IDM statement (HPF), 349
  - IDM statement (HPFIDMSPEC), 243
- NOTSORTED option
  - ID statement (HPF), 345
- NPARMS= option
  - EXM statement (HPFEXMSPEC), 234
- NUM= option
  - ARIMAX statement, 58
  - INPUT statement (HPFARIMASPEC), 30
- NZ= option
  - INPUT statement (HPFARIMASPEC), 30
- OFFSET= option
  - BLOCKSEASON statement (HPFUCMSPEC), 311
- OPERATION= option
  - ADJUST statement, 57
  - ADJUST statement (ENG), 119
- OUT= option
  - EVENTDATA statement, 196
  - PROC HPF statement, 335
  - PROC HPFENGINE statement, 111
  - PROC HPFEVENTS statement, 199
- OUTCOMPONENT= option
  - PROC HPFENGINE statement, 111
- OUTEST= option
  - PROC HPF statement, 335
  - PROC HPFDIAGNOSE statement, 55
  - PROC HPFENGINE statement, 111
- OUTFOR= option
  - PROC HPF statement, 335
  - PROC HPFENGINE statement, 111
  - PROC HPFRECONCILE statement, 254
- OUTINDEP= option
  - PROC HPFENGINE statement, 111
- OUTINFEASIBLE= option
  - PROC HPFRECONCILE statement, 254
- OUTLIER= option
  - ARIMAX statement, 59
- OUTMODELINFO= option
  - PROC HPFENGINE statement, 111
- OUTOUTLIER= option
  - PROC HPFDIAGNOSE statement, 55
- OUTPROCINFO= option
  - PROC HPF statement, 335
  - PROC HPFENGINE statement, 112
  - PROC HPFDIAGNOSE statement, 55
  - PROC HPFRECONCILE statement, 255
- OUTSEASON= option
  - PROC HPF statement, 335
- OUTSTAT= option
  - PROC HPF statement, 335
  - PROC HPFENGINE statement, 111
- OUTSTATSELECT= option
  - PROC HPFENGINE statement, 111
- OUTSUM= option
  - PROC HPF statement, 335
- OUTTREND= option



- PROC HPF statement, 336
- P= option
  - ARIMAX statement, 58
  - FORECAST statement (HPFARIMASPEC), 27
  - TRANSFORM statement, 66
  - TREND statement, 68
- PERIOD= option
  - CYCLE statement (HPFUCMSPEC), 312
  - EVENTDEF statement (HPFEVENTS), 197
- PERROR= option
  - ARIMAX statement, 58
- PHI= option
  - DEPLAG statement (HPFUCMSPEC), 314
- PLOT= option
  - PROC HPF statement, 336
  - PROC HPFENGINE statement, 112
- PREDEFINED= option
  - INPUT statement (HPFARIMASPEC), 29
  - INPUT statement (HPFUCMSPEC), 315
- PREDICTONLY option
  - PROC HPFRECONCILE statement, 256
- PREFILTER= option
  - PROC HPFDIAGNOSE statement, 55
- PRINT= option
  - PROC HPF statement, 336
  - PROC HPFDIAGNOSE statement, 55
  - PROC HPFENGINE statement, 112
- PRINTDETAILS option
  - PROC HPF statement, 337
  - PROC HPFENGINE statement, 113
- PROC HPF statement, 334
- PROC HPFARIMASPEC statement, 26
- PROC HPFDIAG statement, 50
- PROC HPFDIAGNOSE statement, 47, 50
- PROC HPFENGINE statement, 110
- PROC HPFESMSPEC statement, 177
- PROC HPFEVENTS statement, 195
- PROC HPFEXMSPEC statement, 233
- PROC HPFIDMSPEC statement, 239
- PROC HPFRECONCILE statement, 254
- PROC HPFSELECT statement, 284
- PROC HPFUCMSPEC statement, 309
- PULSE= option
  - EVENTDEF statement (HPFEVENTS), 198
- Q= option
  - ARIMAX statement, 58
  - FORECAST statement (HPFARIMASPEC), 28
- RECDIFF= option
  - PROC HPFRECONCILE statement, 255
- REFINEPARMS= option
  - ARIMAX statement, 60
  - UCM statement, 69
- REPLACEBACK option
  - FORECAST statement (HPF), 341
- REPLACEMISSING option
  - FORECAST statement (ENG), 123
  - FORECAST statement (HPF), 341
  - STOCHASTIC statement (ENG), 129
- REPOSITORY= option
  - PROC HPFDIAGNOSE statement, 55
  - PROC HPFENGINE statement, 113
  - PROC HPFESMSPEC statement, 177
  - PROC HPFEXMSPEC statement, 233
  - PROC HPFIDMSPEC statement, 239
  - PROC HPFSELECT statement, 285
- REQUIRED=
  - INPUT statement (ENG), 127
  - STOCHASTIC statement (ENG), 129
- REQUIRED= option
  - EVENT statement, 62
  - INPUT statement, 65
- RETAINCHOOSE option
  - PROC HPFDIAGNOSE statement, 55
- RHO= option
  - AUTOREG statement (HPFUCMSPEC), 310
  - CYCLE statement (HPFUCMSPEC), 313
- RULE option
  - EVENTCOMB statement (HPFEVENTS), 196
  - EVENTDEF statement (HPFEVENTS), 198
- SCORE statement
  - HPFENGINE procedure, 128
- SCOREREPOSITORY= option
  - PROC HPFENGINE statement, 113
- SDIFF= option
  - TREND statement, 67
- SEASON statement
  - HPFUCMSPEC procedure, 317
- SEASONALITY= number
  - PROC HPFENGINE statement, 113
- SEASONALITY= option
  - PROC HPF statement, 337
  - PROC HPFDIAGNOSE statement, 56
- SEASONPARM= option
  - ESM statement (HPFESMSPEC), 180
- SEASONREST= option
  - ESM statement (HPFESMSPEC), 181
- SEASONTTEST= option
  - DIAGNOSE statement (HPFSELECT), 286
  - FORECAST statement (HPF), 341
- SELECT statement
  - HPFSELECT procedure, 286

- SELECT= option
  - ESM statement (HPFESMSPEC), 178
  - FORECAST statement (HPF), 338
  - IDM statement (HPF), 350
  - IDM statement (HPFIDMSPEC), 241
- SELECTBASE= option
  - PROC HPFDIAGNOSE statement, 56
- SELECTEVENT= option
  - PROC HPFDIAGNOSE statement, 56
- SELECTINPUT= option
  - PROC HPFDIAGNOSE statement, 56
- SELECTION=
  - STOCHASTIC statement (ENG), 129
- SETMISSINF= option
  - INPUT statement (ENG), 128
- SETMISSING= option
  - ADJUST statement, 58
  - ADJUST statement (ENG), 120
  - CONTROL statement (ENG), 121
  - EXTERNAL statement (ENG), 122
  - FORECAST statement, 62
  - FORECAST statement (ENG), 123
  - FORECAST statement (HPF), 342
  - ID statement, 63
  - ID statement (HPF), 345
  - ID statement (HPFENGINE), 125
  - ID statement (HPFEVENTS), 202
  - INPUT statement, 66
  - STOCHASTIC statement (ENG), 130
- SHIFT= option
  - EVENTDEF statement (HPFEVENTS), 198
- SIGLEVEL= option
  - ARIMAX statement, 59
  - PROC HPFDIAGNOSE statement, 56
  - TRANSFORM statement, 66
  - TREND statement, 68
  - UCM statement, 69
- SIGMA= option
  - EXM statement (HPFEXMSPEC), 234
- SIGN= option
  - PROC HPFRECONCILE statement, 256
- SINGULAR= option
  - ESTIMATE statement (HPFARIMASPEC), 32
- SIZE= option
  - IDM statement (HPF), 348
  - IDM statement (HPFIDMSPEC), 240
- SLOPE statement
  - HPFUCMSPEC procedure, 319
- SLOPE= option
  - EVENTDEF statement (HPFEVENTS), 197
- SORTNAMES option
  - PROC HPF statement, 337
  - PROC HPFENGINE statement, 113
- PROC HPFEVENTS statement, 195
- SPECBASE= option
  - PROC HPFDIAGNOSE statement, 56
- SPECIFICATION statement
  - HPFSELECT procedure, 288
- SPECLABEL= option
  - PROC HPFESMSPEC statement, 177
- SPECNAME= option
  - PROC HPFESMSPEC statement, 177
- Standard Errors
  - Bottom-up Reconciliation (HPFRECONCILE), 265
  - Top-Down Reconciliation (HPFRECONCILE), 264
- START= option
  - ID statement, 64
  - ID statement (ENG), 126
  - ID statement (HPF), 345
  - ID statement (HPFEVENTS), 203
  - ID statement (HPFRECONCILE), 259
- STARTSUM= option
  - PROC HPF statement, 337
- STDDIFBD= option
  - PROC HPFRECONCILE statement, 257
- STDMETHOD= option
  - PROC HPFRECONCILE statement, 256
- STOCHASTIC statement
  - HPFENGINE procedure, 128
- TASK= option
  - PROC HPFENGINE statement (ENG), 113
- TCPARM= option
  - EVENTDEF statement (HPFEVENTS), 199
- TESTINPUT= option
  - PROC HPFDIAGNOSE statement, 56
- Top-Down Reconciliation
  - HPFRECONCILE procedure, 261
- TRANSFORM statement
  - HPFDIAGNOSE procedure, 66
- TRANSFORM= option
  - ESM statement (HPFESMSPEC), 181
  - EXM statement (HPFEXMSPEC), 234
  - FORECAST statement (HPF), 342
  - FORECAST statement (HPFARIMASPEC), 28
  - IDM statement (HPF), 350
  - IDM statement (HPFIDMSPEC), 243
  - INPUT statement (HPFARIMASPEC), 30
  - INPUT statement (HPFUCMSPEC), 314, 315
- TRANSOPT= option
  - FORECAST statement (HPFARIMASPEC), 28
- TREND statement

- HPFDIAGNOSE procedure, 67
- TRENDPARM= option
  - ESM statement (HPFESMSPEC), 181
  - IDM statement (HPF), 350
  - IDM statement (HPFIDMSPEC), 244
- TRENDREST= option
  - ESM statement (HPFESMSPEC), 181
  - IDM statement (HPF), 350
  - IDM statement (HPFIDMSPEC), 244
- TRIMMISS= option
  - ADJUST statement, 58
  - ADJUST statement (ENG), 120
  - CONTROL statement (ENG), 121
  - EXTERNAL statement (ENG), 122
  - FORECAST statement, 62
  - FORECAST statement (ENG), 123, 126
  - ID statement, 64
  - INPUT statement, 66
  - INPUT statement (ENG), 128
  - STOCHASTIC statement (ENG), 130
- TYPE= option
  - BLOCKSEASON statement (HPFUCMSPEC), 311
  - EVENTDEF statement (HPFEVENTS), 199
  - SEASON statement (HPFUCMSPEC), 318
  - TRANSFORM statement, 66
- UCM statement
  - HPFDIAGNOSE procedure, 68
- USE= option
  - FORECAST statement (HPF), 342
- VALIDATE option
  - PROC HPFSELECT statement, 285
- VALUE= option
  - EVENTDEF statement (HPFEVENTS), 199
- VAR statement
  - HPFEVENTS procedure, 203
- VARIANCE= option
  - AUTOREG statement (HPFUCMSPEC), 310
  - BLOCKSEASON statement (HPFUCMSPEC), 311
  - CYCLE statement (HPFUCMSPEC), 313
  - IRREGULAR statement (HPFUCMSPEC), 316
  - LEVEL statement (HPFUCMSPEC), 317
  - SEASON statement (HPFUCMSPEC), 318
  - SLOPE statement (HPFUCMSPEC), 319
- WEIGHTED option
  - PROC HPFRECONCILE statement, 257
- ZEROMISS= option
  - ADJUST statement, 58
- ADJUST statement (ENG), 120
- CONTROL statement (ENG), 121
- EXTERNAL statement (ENG), 122
- FORECAST statement, 62
- FORECAST statement (ENG), 123
- FORECAST statement (HPF), 342
- ID statement, 64
- INPUT statement, 66
- INPUT statement (ENG), 128
- STOCHASTIC statement (ENG), 130
- ZEROMISSING= option
  - ID statement (ENG), 126
  - ID statement (PROC HPF), 346



# Your Turn

---

We want your feedback.

- If you have comments about this book, please send them to **[yourturn@sas.com](mailto:yourturn@sas.com)**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **[suggest@sas.com](mailto:suggest@sas.com)**.



# SAS® Publishing gives you the tools to flourish in any environment with SAS!

Whether you are new to the workforce or an experienced professional, you need to distinguish yourself in this rapidly changing and competitive job market. SAS® Publishing provides you with a wide range of resources to help you set yourself apart.

## SAS® Press Series

Need to learn the basics? Struggling with a programming problem? You'll find the expert answers that you need in example-rich books from the SAS Press Series. Written by experienced SAS professionals from around the world, these books deliver real-world insights on a broad range of topics for all skill levels.

[support.sas.com/saspress](http://support.sas.com/saspress)

## SAS® Documentation

To successfully implement applications using SAS software, companies in every industry and on every continent all turn to the one source for accurate, timely, and reliable information—SAS documentation. We currently produce the following types of reference documentation: online help that is built into the software, tutorials that are integrated into the product, reference documentation delivered in HTML and PDF—free on the Web, and hard-copy books.

[support.sas.com/publishing](http://support.sas.com/publishing)

## SAS® Learning Edition 4.1

Get a workplace advantage, perform analytics in less time, and prepare for the SAS Base Programming exam and SAS Advanced Programming exam with SAS® Learning Edition 4.1. This inexpensive, intuitive personal learning version of SAS includes Base SAS® 9.1.3, SAS/STAT®, SAS/GRAPH®, SAS/QC®, SAS/ETS®, and SAS® Enterprise Guide® 4.1. Whether you are a professor, student, or business professional, this is a great way to learn SAS.

[support.sas.com/LE](http://support.sas.com/LE)



**THE  
POWER  
TO KNOW®**

