SAS/ETS® 12.3 User's Guide High-Performance Procedures

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2013. *SAS/ETS® 12.3 User's Guide: High-Performance Procedures*. Cary, NC: SAS Institute Inc.

SAS/ETS® 12.3 User's Guide: High-Performance Procedures

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

July 2013

Contents

Credits and Acknowledgments

Credits

Documentation

Editing	Anne Baxter, Ed Huddleston
Documentation Support	Tim Arnold

Software

The procedures in this book were was implemented by the following members of the development staff. Program development includes design, programming, debugging, support, and documentation. In the following list, the names of the developers who currently provide primary support are listed first; other developers and previous developers are also listed.

HPCOUNTREG	Richard Potter, Jan Chvosta
HPQLIM	Richard Potter, Christian Macaro, Jan Chvosta
HPSEVERITY	Mahesh Joshi
High-performance computing foundation	Steve E. Krueger
High-performance analytics foundation	Robert Cohen, Georges H. Guirguis, Trevor Kearney, Richard Knight, Gang Meng, Oliver Schabenberger, Charles Shorb, Tom P. Weber
Numerical routines	Georges H. Guirguis

The following people contribute with their leadership and support: Chris Bailey, Tanya Balan, David Pope, Oliver Schabenberger, Renee Sciortino.

Testing

Ming Chun-Chang, Bruce Elsheimer, Sanggohn Han, Oleksiy Tokovenko, Jim McKenzie, Dright Ho, Girija Gavankar, Tim Carter, Bengt Pederson, Cheryl LeSaint, Jim Metcalf.

Internationalization Testing

Alex Chai, Mi-Na Chu, Jacky Dong, Feng Gao, Masayuki Iizuka, David Li, Lan Luan, Haiyong Rong, Bin Sun, Frank Wang, Lina Xu, Catherine Yang.

Technical Support

Phil Gibbs

Acknowledgments

Many people make significant and continuing contributions to the development of SAS software products.

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

Chapter 1
Introduction

Contents

Overview of SAS/ETS High-Performance Procedures

SAS/ETS high-performance procedures provide econometric modeling tools that have been specially developed to take advantage of parallel processing in both multithreaded single-machine mode and distributed multiple-machine mode. Econometric modeling methods include regression for count data, models for the severity of losses or other events, and regression models for qualitative and limited dependent variables.

In addition to the high-performance econometric procedures described in this book, SAS/ETS includes high-performance utility procedures, which are described in *Base SAS Procedures Guide: High-Performance Procedures*. You can run all these procedures in single-machine mode without licensing SAS High-Performance Econometrics. However, to run these procedures in distributed mode, you must license SAS High-Performance Econometrics.

About This Book

This book assumes that you are familiar with Base SAS software and with the books *SAS Language Reference: Concepts* and *Base SAS Procedures Guide*. It also assumes that you are familiar with basic SAS System concepts, such as using the DATA step to create SAS data sets and using Base SAS procedures (such as, the PRINT and SORT procedures) to manipulate SAS data sets.

Chapter Organization

This book is organized as follows:

Chapter 1, this chapter, provides an overview of SAS/ETS high-performance procedures.

Chapter 2, "Shared Concepts and Topics," describes the modes in which SAS/ETS high-performance procedures can execute.

Subsequent chapters describe the individual procedures. These chapters appear in alphabetical order by procedure name. Each chapter is organized as follows:

- The "Overview" section provides a brief description of the analysis provided by the procedure.
- The "Getting Started" section provides a quick introduction to the procedure through a simple example.
- The "Syntax" section describes the SAS statements and options that control the procedure.
- The "Details" section discusses methodology and other topics, such as ODS tables.
- The "Examples" section contains examples that use the procedure.
- The "References" section contains references for the methodology.

Typographical Conventions

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

roman	is the standard type style used for most text.
UPPERCASE ROMAN	is used for SAS statements, options, and other SAS language elements when they appear in the text. However, you can enter these elements in your own SAS programs in lowercase, uppercase, or a mixture of the two.
UPPERCASE BOLD	is used in the "Syntax" sections' initial lists of SAS statements and options.
oblique	is used in the syntax definitions and in text to represent arguments for which you supply a value.
VariableName	is used for the names of variables and data sets when they appear in the text.
bold	is used to for matrices and vectors.
italic	is used for terms that are defined in the text, for emphasis, and for references to publications.
`monospace`	is used for example code. In most cases, this book uses lowercase type for SAS code.

Options Used in Examples

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

```
options linesize=80 pagesize=500 nonumber nodate;
```

The HTMLBLUE style is used to create the HTML output and graphs that appear in the online documentation. A style template controls stylistic elements such as colors, fonts, and presentation attributes. The style template is specified in the ODS HTML statement as follows:

```
ods html style=HTMLBlue;
```

If you run the examples, your output might be slightly different, because of the SAS System options you use and the precision that your computer uses for floating-point calculations.

Online Documentation

This documentation is available online with the SAS System. To access documentation for the SAS/ETS high-performance procedures from the SAS windowing environment, select **Help** from the main menu and then select **SAS Help and Documentation**. On the **Contents** tab, expand the **SAS Products**, **SAS/ETS**, and **SAS/ETS User's Guide: High-Performance Procedures** items. Then expand chapters and click on sections. You can search the documentation by using the **Search** tab.

You can also access the documentation by going to `http://support.sas.com/documentation`.

SAS Technical Support Services

The SAS Technical Support staff is available to respond to problems and answer technical questions regarding the use of high-performance procedures. Go to `http://support.sas.com/techsup` for more information.

Chapter 2
Shared Concepts and Topics

Contents

Overview

This chapter describes the modes of execution in which SAS high-performance analytical procedures can execute. If you have SAS/ETS installed, you can run any procedure in this book on a single machine.

However, to run procedures in this book in distributed mode, you must also have SAS High-Performance Econometrics software installed. For more information about these modes, see the next section.

This chapter provides details of how you can control the modes of execution and includes the syntax for the PERFORMANCE statement, which is common to all high-performance analytical procedures.

Processing Modes

Single-Machine Mode

Single-machine mode is a computing model in which multiple processors or multiple cores are controlled by a single operating system and can access shared resources, such as disks and memory. In this book, single-machine mode refers to an application running multiple concurrent threads on a multicore machine in order to take advantage of parallel execution on multiple processing units. More simply, single-machine mode for high-performance analytical procedures means multithreading on the client machine.

All high-performance analytical procedures are capable of running in single-machine mode, and this is the default mode when a procedure runs on the client machine. The procedure uses the number of CPUs (cores) on the machine to determine the number of concurrent threads. High-performance analytical procedures use different methods to map core count to the number of concurrent threads, depending on the analytic task. Using one thread per core is not uncommon for the procedures that implement data-parallel algorithms.

Distributed Mode

Distributed mode is a computing model in which several nodes in a distributed computing environment participate in the calculations. In this book, the distributed mode of a high-performance analytical procedure refers to the procedure performing the analytics on an appliance that consists of a cluster of nodes. This appliance can be one of the following:

- a database management system (DBMS) appliance on which the SAS High-Performance Analytics infrastructure is also installed
- a cluster of nodes that have the SAS High-Performance Analytics infrastructure installed but no DBMS software installed

Distributed mode has several variations:

- Client-data (or local-data) mode: The input data for the analytic task are not stored on the appliance or cluster but are distributed to the distributed computing environment by the SAS High-Performance Analytics infrastructure when the procedure runs.
- Alongside-the-database mode: The data are stored in the distributed database and are read from the DBMS in parallel into a high-performance analytical procedure that runs on the database appliance.

- Alongside-HDFS mode: The data are stored in the Hadoop Distributed File System (HDFS) and are read in parallel from the HDFS. This mode is available if you install the SAS High-Performance Deployment of Hadoop on the appliance or when you configure a Cloudera 4 Hadoop deployment on the appliance to operate with the SAS High-Performance Analytics infrastructure. For more information about installing the SAS High-Performance Deployment of Hadoop, see the *SAS High-Performance Analytics Infrastructure: Installation and Configuration Guide*.

- Alongside-LASR mode: The data are loaded from a SAS LASR Analytic Server that runs on the appliance.

Symmetric and Asymmetric Distributed Modes

SAS high-performance analytical procedures can run alongside the database or alongside HDFS in asymmetric mode. The primary reason for providing the asymmetric mode is to enable you to manage and house data on one appliance (the data appliance) and to run the high-performance analytical procedure on a second appliance (the computing appliance). You can also run in asymmetric mode on a single appliance that functions as both the data appliance and the computing appliance. This enables you to run alongside the database or alongside HDFS, where computations are done on a different set of nodes from the nodes that contain the data. The following subsections provide more details.

Symmetric Mode

When SAS high-performance analytical procedures run in symmetric distributed mode, the data appliance and the computing appliance must be the same appliance. Both the SAS Embedded Process and the high-performance analytical procedures execute in a SAS process that runs on the same hardware where the DBMS process executes. This is called symmetric mode because the number of nodes on which the DBMS executes is the same as the number of nodes on which the high-performance analytical procedures execute. The initial data movement from the DBMS to the high-performance analytical procedure does not cross node boundaries.

Asymmetric Mode

When SAS high-performance analytical procedures run in asymmetric distributed mode, the data appliance and computing appliance are usually distinct appliances. The high-performance analytical procedures execute in a SAS process that runs on the computing appliance. The DBMS and a SAS Embedded Process run on the data appliance. Data are requested by a SAS data feeder that runs on the computing appliance and communicates with the SAS Embedded Process on the data appliance. The SAS Embedded Process transfers the data in parallel to the SAS data feeder that runs on each of the nodes of the computing appliance. This is called asymmetric mode because the number of nodes on the data appliance does not need to be the same as the number of nodes on the computing appliance.

Controlling the Execution Mode with Environment Variables and Performance Statement Options

You control the execution mode by using environment variables or by specifying options in the PERFORMANCE statement in high-performance analytical procedures, or by a combination of these methods.

The important environment variables follow:

- *grid host* identifies the domain name system (DNS) or IP address of the appliance node to which the SAS High-Performance Econometrics software connects to run in distributed mode.
- *installation location* identifies the directory where the SAS High-Performance Econometrics software is installed on the appliance.
- *data server* identifies the database server on Teradata appliances as defined in the *hosts* file on the client. This data server is the same entry that you usually specify in the SERVER= entry of a LIBNAME statement for Teradata. For more information about specifying LIBNAME statements for Teradata and other engines, see the DBMS-specific section of *SAS/ACCESS for Relational Databases: Reference* for your engine.
- *grid mode* specifies whether the high-performance analytical procedures execute in symmetric or asymmetric mode. Valid values for this variable are `'sym'` for symmetric mode and `'asym'` for asymmetric mode. The default is symmetric mode.

You can set an environment variable directly from the SAS program by using the OPTION SET= command. For example, the following statements define three variables for a Teradata appliance (the grid mode is the default symmetric mode):

```
option set=GRIDHOST      ="hpa.sas.com";
option set=GRIDINSTALLLOC="/opt/TKGrid";
option set=GRIDDATASERVER="myserver";
```

Alternatively, you can set the parameters in the PERFORMANCE statement in high-performance analytical procedures. For example:

```
performance host      ="hpa.sas.com"
            install   ="/opt/TKGrid"
            dataserver="myserver";
```

The following statements define three variables that are needed to run asymmetrically on a computing appliance.

```
option set=GRIDHOST      ="compute_appliance.sas.com";
option set=GRIDINSTALLLOC="/opt/TKGrid";
option set=GRIDMODE      ="asym";
```

Alternatively, you can set the parameters in the PERFORMANCE statement in high-performance analytical procedures. For example:

```
performance host      ="compute_appliance.sas.com"
            install   ="/opt/TKGrid"
            gridmode  ="asym"
```

A specification in the PERFORMANCE statement overrides a specification of an environment variable without resetting its value. An environment variable that you set in the SAS session by using an OPTION SET= command remains in effect until it is modified or until the SAS session terminates.

Specifying a data server is necessary only on Teradata systems when you do not explicitly set the *gridmode* environment variable or specify the GRIDMODE= option in the PERFORMANCE statement. The data server specification depends on the entries in the (client) *hosts* file. The file specifies the server (suffixed by *cop* and a number) and an IP address. For example:

```
myservercop1  33.44.55.66
```

The key variable that determines whether a high-performance analytical procedure executes in single-machine or distributed mode is the *grid host*. The installation location and data server are needed to ensure that a connection to the grid host can be made, given that a host is specified. This book assumes that the installation location and data server (if necessary) have been set by your system administrator.

The following sets of SAS statements are functionally equivalent:

```
proc hpreduce;
   reduce unsupervised x:;
   performance host="hpa.sas.com";
run;

option set=GRIDHOST="hpa.sas.com";
proc hpreduce;
   reduce unsupervised x:;
run;
```

Determining Single-Machine Mode or Distributed Mode

High-performance analytical procedures use the following rules to determine whether they run in single-machine mode or distributed mode:

- If a grid host is not specified, the analysis is carried out in single-machine mode on the client machine that runs the SAS session.
- If a grid host is specified, the behavior depends on whether the execution is alongside the database or alongside HDFS. If the data are local to the client (that is, not stored in the distributed database or HDFS on the appliance), you need to use the NODES= option in the PERFORMANCE statement to specify the number of nodes on the appliance or cluster that you want to engage in the analysis. If the procedure executes alongside the database or alongside HDFS, you do not need to specify the NODES= option.

The following example shows single-machine and client-data distributed configurations for a data set of 100,000 observations that are simulated from a logistic regression model. The following DATA step generates the data:

```
data simData;
   array _a{8} _temporary_ (0,0,0,1,0,1,1,1);
   array _b{8} _temporary_ (0,0,1,0,1,0,1,1);
   array _c{8} _temporary_ (0,1,0,0,1,1,0,1);
   do obsno=1 to 100000;
      x  = rantbl(1,0.28,0.18,0.14,0.14,0.03,0.09,0.08,0.06);
      a  = _a{x};
      b  = _b{x};
      c  = _c{x};
      x1 = int(ranuni(1)*400);
      x2 = 52 + ranuni(1)*38;
      x3 = ranuni(1)*12;
      lp = 6. -0.015*(1-a) + 0.7*(1-b) + 0.6*(1-c) + 0.02*x1 -0.05*x2 - 0.1*x3;
      y  = ranbin(1,1,(1/(1+exp(lp))));
      output;
   end;
   drop x lp;
run;
```

The following statements run PROC HPLOGISTIC to fit a logistic regression model:

```
proc hplogistic data=simData;
   class a b c;
   model y = a b c x1 x2 x3;
run;
```

Figure 2.1 shows the results from the analysis.

Figure 2.1 Results from Logistic Regression in Single-Machine Mode

```
                              The HPLOGISTIC Procedure

                              Performance Information

                        Execution Mode       Single-Machine
                        Number of Threads    4


                                 Model Information

               Data Source                 WORK.SIMDATA
               Response Variable           y
               Class Parameterization      GLM
               Distribution                Binary
               Link Function               Logit
               Optimization Technique      Newton-Raphson with Ridging
```

Figure 2.1 *continued*

```
                              Parameter Estimates

                                   Standard
        Parameter    Estimate         Error        DF    t Value    Pr > |t|

        Intercept      5.7011        0.2539     Infty      22.45      <.0001
        a 0          -0.01020       0.06627     Infty      -0.15      0.8777
        a 1                 0             .         .          .           .
        b 0            0.7124       0.06558     Infty      10.86      <.0001
        b 1                 0             .         .          .           .
        c 0            0.8036       0.06456     Infty      12.45      <.0001
        c 1                 0             .         .          .           .
        x1            0.01975      0.000614     Infty      32.15      <.0001
        x2           -0.04728      0.003098     Infty     -15.26      <.0001
        x3            -0.1017      0.009470     Infty     -10.74      <.0001
```

The entries in the “Performance Information” table show that the HPLOGISTIC procedure runs in single-machine mode and uses four threads, which are chosen according to the number of CPUs on the client machine. You can force a certain number of threads on any machine that is involved in the computations by specifying the NTHREADS option in the PERFORMANCE statement. Another indication of execution on the client is the following message, which is issued in the SAS log by all high-performance analytical procedures:

```
NOTE: The HPLOGISTIC procedure is executing on the client.
```

The following statements use 10 nodes (in distributed mode) to analyze the data on the appliance; results appear in Figure 2.2:

```
proc hplogistic data=simData;
   class a b c;
   model y = a b c x1 x2 x3;
   performance host="hpa.sas.com" nodes=10;
run;
```

Figure 2.2 Results from Logistic Regression in Distributed Mode

```
                             The HPLOGISTIC Procedure

                             Performance Information

                    Host Node                        hpa.sas.com
                    Execution Mode                   Distributed
                    Grid Mode                        Symmetric
                    Number of Compute Nodes          10
                    Number of Threads per Node       24
```

Figure 2.2 *continued*

```
                              Model Information

              Data Source                 WORK.SIMDATA
              Response Variable           y
              Class Parameterization      GLM
              Distribution                Binary
              Link Function               Logit
              Optimization Technique      Newton-Raphson with Ridging


                             Parameter Estimates

                                  Standard
        Parameter    Estimate        Error        DF    t Value    Pr > |t|

        Intercept      5.7011       0.2539    Infty      22.45      <.0001
        a 0          -0.01020      0.06627    Infty      -0.15      0.8777
        a 1                 0            .        .          .           .
        b 0            0.7124      0.06558    Infty      10.86      <.0001
        b 1                 0            .        .          .           .
        c 0            0.8036      0.06456    Infty      12.45      <.0001
        c 1                 0            .        .          .           .
        x1            0.01975     0.000614    Infty      32.15      <.0001
        x2           -0.04728     0.003098    Infty     -15.26      <.0001
        x3            -0.1017     0.009470    Infty     -10.74      <.0001
```

The specification of a host causes the “Performance Information” table to display the name of the host node of the appliance. The “Performance Information” table also indicates that the calculations were performed in a distributed environment on the appliance. Twenty-four threads on each of 10 nodes were used to perform the calculations—for a total of 240 threads.

Another indication of distributed execution on the appliance is the following message, which is issued in the SAS log by all high-performance analytical procedures:

```
NOTE: The HPLOGISTIC procedure is executing in the distributed
      computing environment with 10 worker nodes.
```

You can override the presence of a grid host and force the computations into single-machine mode by specifying the NODES=0 option in the PERFORMANCE statement:

```
proc hplogistic data=simData;
   class a b c;
   model y = a b c x1 x2 x3;
   performance host="hpa.sas.com" nodes=0;
run;
```

Figure 2.3 shows the “Performance Information” table. The numeric results are not reproduced here, but they agree with the previous analyses, which are shown in Figure 2.1 and Figure 2.2.

Figure 2.3 Single-Machine Mode Despite Host Specification

```
              The HPLOGISTIC Procedure

              Performance Information

        Execution Mode       Single-Machine
        Number of Threads    4
```

The “Performance Information” table indicates that the HPLOGISTIC procedure executes in single-machine mode on the client. This information is also reported in the following message, which is issued in the SAS log:

```
NOTE: The HPLOGISTIC procedure is executing on the client.
```

In the analysis shown previously in Figure 2.2, the data set Work.simData is local to the client, and the HPLOGISTIC procedure distributed the data to 10 nodes on the appliance. The High-Performance Analytics infrastructure does not keep these data on the appliance. When the procedure terminates, the in-memory representation of the input data on the appliance is freed.

When the input data set is large, the time that is spent sending client-side data to the appliance might dominate the execution time. In practice, transfer speeds are usually lower than the theoretical limits of the network connection or disk I/O rates. At a transfer rate of 40 megabytes per second, sending a 10-gigabyte data set to the appliance requires more than four minutes. If analytic execution time is in the range of seconds, the “performance” of the process is dominated by data movement.

The alongside-the-database execution model, unique to high-performance analytical procedures, enables you to read and write data in distributed form from the database that is installed on the appliance.

Alongside-the-Database Execution

High-performance analytical procedures interface with the distributed database management system (DBMS) on the appliance in a unique way. If the input data are stored in the DBMS and the grid host is the appliance that houses the data, high-performance analytical procedures create a distributed computing environment in which an analytic process is co-located with the nodes of the DBMS. Data then pass from the DBMS to the analytic process on each node. Instead of moving across the network and possibly back to the client machine, the data pass locally between the processes on each node of the appliance.

Because the analytic processes on the appliance are separate from the database processes, the technique is referred to as alongside-the-database execution in contrast to in-database execution, where the analytic code executes in the database process.

In general, when you have a large amount of input data, you can achieve the best performance from high-performance analytical procedures if execution is alongside the database.

Before you can run alongside the database, you must distribute the data to the appliance. The following statements use the HPDS2 procedure to distribute the data set Work.simData into the mydb database on the hpa.sas.com appliance. In this example, the appliance houses a Greenplum database.

```
option set=GRIDHOST="hpa.sas.com";
libname applianc greenplm
        server  ="hpa.sas.com"
        user    =XXXXXX
        password=YYYYY
        database=mydb;


proc datasets lib=applianc nolist; delete simData;
proc hpds2 data=simData
           out =applianc.simData(distributed_by='distributed randomly');
  performance commit=10000 nodes=all;
  data DS2GTF.out;
     method run();
        set DS2GTF.in;
     end;
  enddata;
run;
```

If the output table applianc.simData exists, the DATASETS procedure removes the table from the Greenplum database because a DBMS does not usually support replacement operations on tables.

Note that the libref for the output table points to the appliance. The data set option informs the HPDS2 procedure to distribute the records randomly among the data segments of the appliance. The statements that follow the PERFORMANCE statement are the DS2 program that copies the input data to the output data without further transformations.

Because you loaded the data into a database on the appliance, you can use the following HPLOGISTIC statements to perform the analysis on the appliance in the alongside-the-database mode. These statements are almost identical to the first PROC HPLOGISTIC example in the previous section, which executed in single-machine mode.

```
proc hplogistic data=applianc.simData;
   class a b c;
   model y = a b c x1 x2 x3;
run;
```

The subtle differences are as follows:

- The grid host environment variable that you specified in an OPTION SET= command is still in effect.
- The DATA= option in the high-performance analytical procedure uses a libref that identifies the data source as being housed on the appliance. This libref was specified in a prior LIBNAME statement.

Figure 2.4 shows the results from this analysis. The "Performance Information" table shows that the execution was in distributed mode. In this case the execution was alongside the Greenplum database. The numeric results agree with the previous analyses, which are shown in Figure 2.1 and Figure 2.2.

Figure 2.4 Alongside-the-Database Execution on Greenplum

```
                    The HPLOGISTIC Procedure

                    Performance Information

         Host Node                       hpa.sas.com
         Execution Mode                  Distributed
         Grid Mode                       Symmetric
         Number of Compute Nodes         8
         Number of Threads per Node      24


                       Model Information

   Data Source                 SIMDATA
   Response Variable           y
   Class Parameterization      GLM
   Distribution                Binary
   Link Function               Logit
   Optimization Technique      Newton-Raphson with Ridging
```

Figure 2.4 *continued*

```
                        Parameter Estimates

                          Standard
Parameter     Estimate       Error       DF    t Value    Pr > |t|

Intercept       5.7011      0.2539    Infty      22.45      <.0001
a 0           -0.01020     0.06627    Infty      -0.15      0.8777
a 1                  0           .        .          .           .
b 0             0.7124     0.06558    Infty      10.86      <.0001
b 1                  0           .        .          .           .
c 0             0.8036     0.06456    Infty      12.45      <.0001
c 1                  0           .        .          .           .
x1             0.01975    0.000614    Infty      32.15      <.0001
x2            -0.04728    0.003098    Infty     -15.26      <.0001
x3             -0.1017    0.009470    Infty     -10.74      <.0001
```

When high-performance analytical procedures execute symmetrically alongside the database, any nonzero specification of the NODES= option in the PERFORMANCE statement is ignored. If the data are read alongside the database, the number of compute nodes is determined by the layout of the database and cannot be modified. In this example, the appliance contains 16 nodes. (See the “Performance Information” table.)

However, when high-performance analytical procedures execute asymmetrically alongside the database, the number of compute nodes that you specify in the PERFORMANCE statement can differ from the number of nodes across which the data are partitioned. For an example, see the section “Running High-Performance Analytical Procedures in Asymmetric Mode” on page 19.

Alongside-LASR Distributed Execution

You can execute high-performance analytical procedures in distributed mode alongside a SAS LASR Analytic Server. When high-performance analytical procedures execute in this mode, the data are preloaded in distributed form in memory that is managed by a LASR Analytic Server. The data on the nodes of the appliance are accessed in parallel in the process that runs the LASR Analytic Server, and they are transferred to the process where the high-performance analytical procedure runs. In general, each high-performance analytical procedure copies the data to memory that persists only while that procedure executes. Hence, when a high-performance analytical procedure runs alongside a LASR Analytic Server, both the high-performance analytical procedure and the LASR Analytic Server have a copy of the subset of the data that is used by the high-performance analytical procedure. The advantage of running high-performance analytical procedures alongside a LASR Analytic Server (as opposed to running alongside a DBMS table or alongside HDFS) is that the initial transfer of data from the LASR Analytic Server to the high-performance analytical procedure is a memory-to-memory operation that is faster than the disk-to-memory operation when the procedure runs alongside a DBMS or HDFS. When the cost of preloading a table into a LASR Analytic Server is amortized by multiple uses of these data in separate runs of high-performance analytical procedures, using the LASR Analytic Server can result in improved performance.

Running High-Performance Analytical Procedures Alongside a SAS LASR Analytic Server in Distributed Mode

This section provides an example of steps that you can use to start and load data into a SAS LASR Analytic Server instance and then run high-performance analytical procedures alongside this LASR Analytic Server instance.

Starting a SAS LASR Analytic Server Instance

The following statements create a SAS LASR Analytic Server instance and load it with the simData data set that is used in the preceding examples. The data that are loaded into the LASR Analytic Server persist in memory across procedure boundaries until these data are explicitly deleted or until the server instance is terminated.

```
proc lasr port=12345
          data=simData
          path="/tmp/";
   performance host="hpa.sas.com" nodes=ALL;
run;
```

The PORT= option specifies a network port number to use. The PATH= option specifies the directory in which the server and table signature files are to be stored. The specified directory must exist on each machine in the cluster. The DATA= option specifies the name of a data set that is loaded into this LASR Analytic Server instance. (You do not need to specify the DATA= option at this time because you can add tables to the LASR Analytic Server instance at any stage of its life.) For more information about starting and using a LASR Analytic Server, see the *SAS LASR Analytic Server: Administration Guide*.

The NODES=ALL option in the PERFORMANCE statement specifies that the LASR Analytic Server run on all the nodes on the appliance. You can start a LASR Analytic Server on a subset of the nodes on an appliance, but this might affect whether high-performance analytical procedures can run alongside the LASR Analytic Server. For more information, see the section "Alongside-LASR Distributed Execution on a Subset of the Appliance Nodes" on page 19.

Figure 2.5 shows the "Performance Information" table, which shows that the LASR procedure executes in distributed mode on 16 nodes.

Figure 2.5 Performance Information

```
                         The LASR Procedure

                      Performance Information

              Host Node                    hpa.sas.com
              Execution Mode               Distributed
              Grid Mode                    Symmetric
              Number of Compute Nodes      8
```

Associating a SAS Libref with the SAS LASR Analytic Server Instance

The following statements use a LIBNAME statement that associates a SAS libref (named MyLasr) with tables on the server instance as follows:

```
libname MyLasr sasiola port=12345;
```

The SASIOLA option requests that the MyLasr libref use the SASIOLA engine, and the PORT= value associates this libref with the appropriate server instance. For more information about creating a libref that uses the SASIOLA engine, see the *SAS LASR Analytic Server: Administration Guide*.

Running a High-Performance Analytical Procedure Alongside the SAS LASR Analytic Server Instance

You can use the MyLasr libref to specify the input data for high-performance analytical procedures. You can also create output data sets in the SAS LASR Analytic Server instance by using this libref to request that the output data set be held in memory by the server instance as follows:

```
proc hplogistic data=MyLasr.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   output out=MyLasr.simulateScores pred=PredictedProbabliity;
run;
```

Because you previously specified the GRIDHOST= environment variable and the input data are held in distributed form in the associated server instance, this PROC HPLOGISTIC step runs in distributed mode alongside the LASR Analytic Server, as indicated in the "Performance Information" table shown in Figure 2.6.

Figure 2.6 Performance Information

```
                 Performance Information

        Host Node                        hpa.sas.com
        Execution Mode                   Distributed
        Grid Mode                        Symmetric
        Number of Compute Nodes          8
        Number of Threads per Node       24
```

The preceding OUTPUT statement creates an output table that is added to the LASR Analytic Server instance. Output data sets do not have to be created in the same server instance that holds the input data. You can use a different LASR Analytic Server instance to hold the output data set. However, in order for the output data to be created in alongside mode, all the nodes that are used by the server instance that holds the input data must also be used by the server instance that holds the output data.

Terminating a SAS LASR Analytic Server Instance

You can continue to run high-performance analytical procedures and add and delete tables from the SAS LASR Analytic Server instance until you terminate the server instance as follows:

```
proc lasr term port=12345;
run;
```

Alongside-LASR Distributed Execution on a Subset of the Appliance Nodes

When you run PROC LASR to start a SAS LASR Analytic Server, you can specify the NODES= option in a PERFORMANCE statement to control how many nodes the LASR Analytic Server executes on. Similarly, a high-performance analytical procedure can execute on a subset of the nodes either because you specify the NODES= option in a PERFORMANCE statement or because you run alongside a DBMS or HDFS with an input data set that is distributed on a subset of the nodes on an appliance. In such situations, if a high-performance analytical procedure uses nodes on which the LASR Analytic Server is not running, then running alongside LASR is not supported. You can avoid this issue by specifying the NODES=ALL in the PERFORMANCE statement when you use PROC LASR to start the LASR Analytic Server.

Running High-Performance Analytical Procedures in Asymmetric Mode

This section provides examples of how you can run high-performance analytical procedures in asymmetric mode. It also includes examples that run in symmetric mode to highlight differences between the modes. For a description of asymmetric mode, see the section "Symmetric and Asymmetric Distributed Modes" on page 7.

Asymmetric mode is commonly used when the data appliance and the computing appliance are distinct appliances. In order to be able to use an appliance as a data provider for high-performance analytical procedures that run in asymmetric mode on another appliance, it is not necessary that SAS High-Performance Econometrics be installed on the data appliance. However, it is essential that a SAS Embedded Process be installed on the data appliance and that SAS High-Performance Econometrics be installed on the computing appliance.

The following examples use a 24-node data appliance named "data_appliance.sas.com," which houses a Teradata DBMS and has a SAS Embedded Process installed. Because SAS High-Performance Econometrics is also installed on this appliance, it can be used to run high-performance analytical procedures in both symmetric and asymmetric modes.

The following statements load the simData data set of the preceding sections onto the data appliance:

```
libname dataLib teradata
        server  ="tera2650"
        user    =XXXXXX
        password=YYYYY
        database=mydb;

data dataLib.simData;
   set simData;
run;
```

NOTE: You can provision the appliance with data even if SAS High-Performance Econometrics software is not installed on the appliance.

The following subsections show how you can run the HPLOGISTIC procedure symmetrically and asymmetrically on a single data appliance and asymmetrically on distinct data and computing appliances.

Running in Symmetric Mode

The following statements run the HPLOGISTIC procedure in symmetric mode on the data appliance:

```
proc hplogistic data=dataLib.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   performance host       = "data_appliance.sas.com"
               nodes      = 10
               gridmode   = sym;
run;
```

Because you explicitly specified the GRIDMODE= option, you do not need to also specify the DATASERVER= option in the PERFORMANCE statement. Figure 2.7 shows the results of this analysis.

Figure 2.7 Alongside-the-Database Execution in Symmetric Mode on Teradata

```
                         The HPLOGISTIC Procedure

                         Performance Information

          Host Node                      data_appliance.sas.com
          Execution Mode                 Distributed
          Grid Mode                      Symmetric
          Number of Compute Nodes        24
          Number of Threads per Node     24
```

Figure 2.7 *continued*

```
                          Model Information

         Data Source                  simData
         Response Variable            y
         Class Parameterization       GLM
         Distribution                 Binary
         Link Function                Logit
         Optimization Technique       Newton-Raphson with Ridging


                         Parameter Estimates

                                    Standard
   Parameter     Estimate          Error        DF     t Value     Pr > |t|

   Intercept       5.7011         0.2539     Infty       22.45       <.0001
   a 0           -0.01020        0.06627     Infty       -0.15       0.8777
   a 1                  0              .         .           .            .
   b 0             0.7124        0.06558     Infty       10.86       <.0001
   b 1                  0              .         .           .            .
   c 0             0.8036        0.06456     Infty       12.45       <.0001
   c 1                  0              .         .           .            .
   x1             0.01975       0.000614     Infty       32.15       <.0001
   x2            -0.04728       0.003098     Infty      -15.26       <.0001
   x3             -0.1017       0.009470     Infty      -10.74       <.0001
```

The “Performance Information” table shows that the execution occurs in symmetric mode on the 24 nodes of the data appliance. In this case, the NODES=10 option in the PERFORMANCE statement is ignored because the number of nodes that are used is determined by the number of nodes across which the data are distributed, as indicated in the following warning message in the SAS log:

```
WARNING: The NODES=10 option in the PERFORMANCE statement is ignored because
         you are running alongside the distributed data source
         DATALIB.simData.DATA. The number of compute nodes is determined by the
         configuration of the distributed DBMS.
```

Running in Asymmetric Mode on One Appliance

You can switch to running the HPLOGISTIC procedure in asymmetric mode by specifying the GRIDMODE=ASYM option in the PERFORMANCE statement as follows:

```
proc hplogistic data=dataLib.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   performance host        = "data_appliance.sas.com"
               nodes       = 10
               gridmode    = asym;
run;
```

Figure 2.8 shows the “Performance Information” table.

Figure 2.8 Alongside Teradata Execution in Asymmetric Mode

```
                            The HPLOGISTIC Procedure

                            Performance Information

              Host Node                        data_appliance.sas.com
              Execution Mode                   Distributed
              Grid Mode                        Asymmetric
              Number of Compute Nodes          10
              Number of Threads per Node       24
```

You can see that now the grid mode is asymmetric. Furthermore, the NODES=10 option that you specified in the PERFORMANCE statement is honored. The data are moved in parallel from the 24 nodes on which the data are stored to the 10 nodes on which the execution occurs. The numeric results are not reproduced here, but they agree with the previous analyses.

Running in Asymmetric Mode on Distinct Appliances

Usually, there is no advantage to executing high-performance analytical procedures in asymmetric mode on one appliance, because data might have to be unnecessarily moved between nodes. The following example demonstrates the more typical use of asymmetric mode. In this example, the specified grid host “compute_appliance.sas.com” is a computing appliance that has 15 compute nodes, and it is a different appliance from the 24-node data appliance “data_appliance.sas.com,” which houses the Teradata DBMS where the data reside.

The advantage of using different computing and data appliances is that the data appliance is not affected by the execution of high-performance analytical procedures except during the initial parallel data transfer. A potential disadvantage of this asymmetric mode of execution is that the performance can be limited by the bandwidth with which data can be moved between the appliances. However, because this data movement takes place in parallel from the nodes of the data appliance to the nodes of the computing appliance, this potential performance bottleneck can be overcome with appropriately provisioned hardware. The following statements show how this is done:

```
proc hplogistic data=dataLib.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   performance host        = "compute_appliance.sas.com"
               gridmode    = asym;
run;
```

Figure 2.9 shows the “Performance Information” table.

Figure 2.9 Asymmetric Mode with Distinct Data and Computing Appliances

```
                              The HPLOGISTIC Procedure

                              Performance Information

                Host Node                        compute_appliance.sas.com
                Execution Mode                   Distributed
                Grid Mode                        Asymmetric
                Number of Compute Nodes          15
                Number of Threads per Node       24
```

PROC HPLOGISTIC ran on the 15 nodes of the computing appliance, even though the data are partitioned across the 24 nodes of the data appliance. The numeric results are not reproduced here, but they agree with the previous analyses shown in Figure 2.1 and Figure 2.2.

Every time you run a high-performance analytical procedure in asymmetric mode that uses different computing and data appliances, data are transferred between these appliances. If you plan to make repeated use of the same data, then it might be advantageous to temporarily persist the data that you need on the computing appliance. One way to persist the data is to store them as a table in a SAS LASR Analytic Server that runs on the computing appliance. By running PROC LASR in asymmetric mode, you can load the data in parallel from the data appliance nodes to the nodes on which the LASR Analytic Server runs on the computing appliance. You can then use a LIBNAME statement that associates a SAS libref with tables on the LASR Analytic Server. The following statements show how you do this:

```
proc lasr port=54321
          data=dataLib.simData
          path="/tmp/";
   performance host     ="compute_appliance.sas.com"
               gridmode = asym;
run;

libname MyLasr sasiola tag="dataLib" port=54321 host="compute_appliance.sas.com" ;
```

Figure 2.10 show the “Performance Information” table.

Figure 2.10 PROC LASR Running in Asymmetric Mode

```
                                 The LASR Procedure

                              Performance Information

                  Host Node                     compute_appliance.sas.com
                  Execution Mode                Distributed
                  Grid Mode                     Asymmetric
                  Number of Compute Nodes       15
```

PROC LASR ran in asymmetric mode on the computing appliance, which has 15 compute nodes. In this mode, the data are loaded in parallel from the 24 data appliance nodes to the 15 compute nodes on the computing appliance. By default, all the nodes on the computing appliance are used. You can use the NODES= option in the PERFORMANCE statement to run the LASR Analytic Server on a subset of the nodes on the computing appliance. If you omit the GRIDMODE=ASYM option from the PERFORMANCE statement, PROC LASR still runs successfully but much less efficiently. The Teradata access engine transfers the simData data set to a temporary table on the client, and the High-Performance Analytics infrastructure then transfers these data from the temporary table on the client to the grid nodes on the computing appliance.

After the data are loaded into a LASR Analytic Server that runs on the computing appliance, you can run high-performance analytical procedures alongside this LASR Analytic Server. Because these procedures run on the same computing appliance where the LASR Analytic Server is running, it is best to run these procedures in symmetric mode, which is the default or can be explicitly specified in the GRIDMODE=SYM option in the PERFORMANCE statement. The following statements provide an example. The OUTPUT statement creates an output data set that is held in memory by the LASR Analytic Server. The data appliance has no role in executing these statements.

```
proc hplogistic data=MyLasr.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   output out=MyLasr.myOutputData pred=myPred;
   performance host = "compute_appliance.sas.com";
run;
```

The following note, which appears in the SAS log, confirms that the output data set is created successfully:

```
NOTE: The table DATALIB.MYOUTPUTDATA has been added to the LASR Analytic Server
      with port 54321. The Libname is MYLASR.
```

You can use the dataLib libref that you used to load the data onto the data appliance to create an output data set on the data appliance. In order for this output to be directly written in parallel from the nodes of the computing appliance to the nodes of the data appliance, you need to run the HPLOGISTIC procedure in asymmetric mode by specifying the GRIDMODE=ASYM option in the PERFORMANCE statement as follows:

```
proc hplogistic data=MyLasr.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   output out=dataLib.myOutputData pred=myPred;
   performance host       = "compute_appliance.sas.com"
               gridmode   = asym;
run;
```

The following note, which appears in the SAS log, confirms that the output data set is created successfully on the data appliance:

```
NOTE: The data set DATALIB.myOutputData has 100000 observations and 1 variables.
```

When you run a high-performance analytical procedure on a computing appliance and either read data from or write data to a different data appliance, it is important to run the high-performance analytical procedures in asymmetric mode so that the Read and Write operations take place in parallel without any movement of data to and from the SAS client. If you omit running the preceding PROC HPLOGISTIC step in asymmetric mode, then the output data set would be created much less efficiently: the output data would be moved sequentially to a temporary table on the client, after which the Teradata access engine sequentially would write this table to the data appliance.

When you no longer need the data in the SAS LASR Analytic Server, you should terminate the server instance as follows:

```
proc lasr term port=54321;
    performance host="compute_appliance.sas.com";
run;
```

If you configured Hadoop on the computing appliance, then you can create output data tables that are stored in the HDFS on the computing appliance. You can do this by using the SASHDAT engine as described in the section "Alongside-HDFS Execution" on page 25.

Alongside-HDFS Execution

Running high-performance analytical procedures alongside HDFS shares many features with running alongside the database. You can execute high-performance analytical procedures alongside HDFS by using either the SASHDAT engine or the Hadoop engine.

You use the SASHDAT engine to read and write data that are stored in HDFS in a proprietary SASHDAT format. In SASHDAT format, metadata that describe the data in the Hadoop files are included with the data. This enables you to access files in SASHDAT format without supplying any additional metadata. Additionally, you can also use the SASHDAT engine to read data in CSV (comma-separated value) format, but you need supply metadata that describe the contents of the CSV data. The SASHDAT engine provides highly optimized access to data in HDFS that are stored in SASHDAT format.

The Hadoop engine reads data that are stored in various formats from HDFS and writes data to HDFS in CSV format. This engine can use metadata that are stored in Hive, which is a data warehouse that supplies metadata about data that are stored in Hadoop files. In addition, this engine can use metadata that you create by using the HDMD procedure.

The following subsections provide details about using the SASHDAT and Hadoop engines to execute high-performance analytical procedures alongside HDFS.

Alongside-HDFS Execution by Using the SASHDAT Engine

If the grid host is a cluster that houses data that have been distributed by using the SASHDAT engine, then high-performance analytical procedures can analyze those data in the alongside-HDFS mode. The procedures use the distributed computing environment in which an analytic process is co-located with the nodes of the cluster. Data then pass from HDFS to the analytic process on each node of the cluster.

Before you can run a procedure alongside HDFS, you must distribute the data to the cluster. The following statements use the SASHDAT engine to distribute to HDFS the simData data set that was used in the previous two sections:

```
option set=GRIDHOST="hpa.sas.com";

libname hdatLib sashdat
        path="/hps";
```

```
    data hdatLib.simData (replace = yes) ;
        set simData;
    run;
```

In this example, the GRIDHOST is a cluster where the SAS Data in HDFS Engine is installed. If a data set that is named simData already exists in the hps directory in HDFS, it is overwritten because the REPLACE=YES data set option is specified. For more information about using this LIBNAME statement, see the section "LIBNAME Statement for the SAS Data in HDFS Engine" in the *SAS LASR Analytic Server: Administration Guide*.

The following HPLOGISTIC procedure statements perform the analysis in alongside-HDFS mode. These statements are almost identical to the PROC HPLOGISTIC example in the previous two sections, which executed in single-machine mode and alongside-the-database distributed mode, respectively.

```
   proc hplogistic data=hdatLib.simData;
      class a b c;
      model y = a b c x1 x2 x3;

   run;
```

Figure 2.11 shows the "Performance Information" table. You see that the procedure ran in distributed mode. The numeric results shown in Figure 2.12 agree with the previous analyses shown in Figure 2.1, Figure 2.2, and Figure 2.4.

Figure 2.11 Alongside-HDFS Execution Performance Information

```
                    Performance Information

          Host Node                        hpa.sas.com
          Execution Mode                   Distributed
          Grid Mode                        Symmetric
          Number of Compute Nodes          206
          Number of Threads per Node       8
```

Figure 2.12 Alongside-HDFS Execution Model Information

```
                       Model Information

    Data Source                  HDATLIB.SIMDATA
    Response Variable            y
    Class Parameterization       GLM
    Distribution                 Binary
    Link Function                Logit
    Optimization Technique       Newton-Raphson with Ridging
```

Figure 2.12 *continued*

```
                          Parameter Estimates

                                  Standard
Parameter     Estimate             Error        DF     t Value     Pr > |t|

Intercept       5.7011            0.2539     Infty       22.45       <.0001
a 0           -0.01020           0.06627     Infty       -0.15       0.8777
a 1                  0                 .         .           .            .
b 0             0.7124           0.06558     Infty       10.86       <.0001
b 1                  0                 .         .           .            .
c 0             0.8036           0.06456     Infty       12.45       <.0001
c 1                  0                 .         .           .            .
x1             0.01975          0.000614     Infty       32.15       <.0001
x2            -0.04728          0.003098     Infty      -15.26       <.0001
x3             -0.1017          0.009470     Infty      -10.74       <.0001
```

Alongside-HDFS Execution by Using the Hadoop Engine

The following LIBNAME statement sets up a libref that you can use to access data that are stored in HDFS and have metadata in Hive:

```
libname hdoopLib hadoop
        server        = "hpa.sas.com"
        user          = XXXXX
        password      = YYYYY
        database      = myDB
        config        = "demo.xml" ;
```

For more information about LIBNAME options available for the Hadoop engine, see the LIBNAME topic in the Hadoop section of *SAS/ACCESS for Relational Databases: Reference*. The configuration file that you specify in the CONFIG= option contains information that is needed to access the Hive server. It also contains information that enables this configuration file to be used to access data in HDFS without using the Hive server. This information can also be used to specify replication factors and block sizes that are used when the engine writes data to HDFS. The following XML shows the contents of the file demo.xml that is used in this example:

```
<configuration>
<property>
  <name>fs.default.name</name>
  <value>hdfs://hpa.sas.com:8020</value>
</property>
<property>
  <name>mapred.job.tracker</name>
  <value>hpa.sas.com:8021</value>
</property>
<property>
    <name>dfs.replication</name>
    <value>1</value>
</property>
<property>
    <name>dfs.block.size</name>
    <value>33554432</value>
</property>
</configuration>
```

The following DATA step uses the Hadoop engine to distribute to HDFS the simData data set that was used in the previous sections. The engine creates metadata for the data set in Hive.

```
data hdoopLib.simData;
    set simData;
run;
```

After you have loaded data or if you are accessing preexisting data in HDFS that have metadata in Hive, you can access this data alongside HDFS by using high-performance analytics procedures. The following HPLOGISTIC procedure statements perform the analysis in alongside-HDFS mode. These statements are similar to the PROC HPLOGISTIC example in the previous sections. However, whenever you use the Hadoop engine, you must execute the analysis in asymmetric mode to cause the execution to occur alongside HDFS.

```
proc hplogistic data=hdoopLib.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   performance host     = "compute_appliance.sas.com"
               gridmode = asym;
run;
```

Figure 2.13 shows the “Performance Information” table. You see that the procedure ran asymmetrically in distributed mode. The numeric results shown in Figure 2.14 agree with the previous analyses.

Figure 2.13 Alongside-HDFS Execution by Using the Hadoop Engine

```
                      The HPLOGISTIC Procedure

                      Performance Information

          Host Node                      compute_appliance.sas.com
          Execution Mode                 Distributed
          Grid Mode                      Asymmetric
          Number of Compute Nodes        15
          Number of Threads per Node     24
```

Figure 2.14 Alongside-HDFS Execution by Using the Hadoop Engine

```
                              Model Information

                 Data Source                HDOOPLIB.SIMDATA
                 Response Variable          y
                 Class Parameterization     GLM
                 Distribution               Binary
                 Link Function              Logit
                 Optimization Technique     Newton-Raphson with Ridging


                             Parameter Estimates

                                   Standard
           Parameter    Estimate      Error       DF    t Value    Pr > |t|

           Intercept      5.7011     0.2539    Infty      22.45      <.0001
           a 0          -0.01020    0.06627    Infty      -0.15      0.8777
           a 1                 0          .        .          .           .
           b 0            0.7124    0.06558    Infty      10.86      <.0001
           b 1                 0          .        .          .           .
           c 0            0.8036    0.06456    Infty      12.45      <.0001
           c 1                 0          .        .          .           .
           x1            0.01975   0.000614    Infty      32.15      <.0001
           x2           -0.04728   0.003098    Infty     -15.26      <.0001
           x3            -0.1017   0.009470    Infty     -10.74      <.0001
```

The Hadoop engine also enables you to access tables in HDFS that are stored in various formats and that are not registered in Hive. You can use the HDMD procedure to generate metadata for tables that are stored in the following file formats:

- delimited text
- fixed-record length binary
- JavaScript Object Notation (JSON)
- sequence files
- XML text

To read any other kind of file in Hadoop, you can write a custom file reader plug-in in Java for use with PROC HDMD. For more information about LIBNAME options available for the Hadoop engine, see the LIBNAME topic in the Hadoop section of *SAS/ACCESS for Relational Databases: Reference*.

The following example shows how you can use PROC HDMD to register metadata for CSV data independently from Hive and then analyze these data by using high-performance analytics procedures. The CSV data in the table csvExample.csv is stored in HDFS in the directory /user/demo/data. Each record in this table consists of the following fields, in the order shown and separated by commas.

1. a string of at most six characters
2. a numeric field with values of 0 or 1
3. a numeric field with real numbers

Suppose you want to fit a logistic regression model to these data, where the second field represents a target variable named Success, the third field represents a regressor named Dose, and the first field represents a classification variable named Group.

The first step is to use PROC HDMD to create metadata that are needed to interpret the table, as in the following statements:

```
libname hdoopLib hadoop
                 server       = "hpa.sas.com"
                 user         = XXXXX
                 password     = YYYYY
                 HDFS_PERMDIR = "/user/demo/data"
                 HDFS_METADIR = "/user/demo/meta"
                 config       = "demo.xml"
                 DBCREATE_TABLE_EXTERNAL=YES;

proc hdmd name=hdoopLib.csvExample data_file='csvExample.csv'
          format=delimited encoding=utf8 sep = ',';

     column Group    char(6);
     column Success  double;
     column Dose     double;
run;
```

The metadata that are created by PROC HDMD for this table are stored in the directory /user/demo/meta that you specified in the HDFS_METADIR = option in the preceding LIBNAME statement. After you create the metadata, you can execute high-performance analytics procedures with these data by using the hdoopLib libref. For example, the following statements fit a logistic regression model to the CSV data that are stored in csvExample.csv table.

```
proc hplogistic data=hdoopLib.csvExample;
    class Group;
    model Success = Dose;
    performance host     = "compute_appliance.sas.com"
                gridmode = asym;
run;
```

Figure 2.15 shows the results of this analysis. You see that the procedure ran asymmetrically in distributed mode. The metadata that you created by using the HDMD procedure have been used successfully in executing this analysis.

Figure 2.15 Alongside-HDFS Execution with CSV Data

```
                          The HPLOGISTIC Procedure

                          Performance Information

          Host Node                       compute_appliance.sas.com
          Execution Mode                  Distributed
          Grid Mode                       Asymmetric
          Number of Compute Nodes         15
          Number of Threads per Node      24


                             Model Information

           Data Source                 GRIDLIB.CSVEXAMPLE
           Response Variable           Success
           Class Parameterization      GLM
           Distribution                Binary
           Link Function               Logit
           Optimization Technique      Newton-Raphson with Ridging


                          Class Level Information

                  Class     Levels     Values

                  Group          3     group1 group2 group3


                Number of Observations Read            1000
                Number of Observations Used            1000


                            Parameter Estimates

                                   Standard
     Parameter     Estimate           Error         DF     t Value     Pr > |t|

     Intercept       0.1243          0.1295      Infty        0.96       0.3371
     Dose           -0.2674          0.2216      Infty       -1.21       0.2277
```

Output Data Sets

In the alongside-the-database mode, the data are read in distributed form, minimizing data movement for best performance. Similarly, when you write output data sets and a high-performance analytical procedure executes in distributed mode, the data can be written in parallel into the database.

For example, in the following statements, the HPLOGISTIC procedure executes in distributed mode by using eight nodes on the appliance to perform the logistic regression on work.simData:

```
proc hplogistic data=simData;
   class a b c;
   model y = a b c x1 x2 x3;
   id a;
   output out=applianc.simData_out pred=p;
   performance host="hpa.sas.com" nodes=8;
run;
```

The output data set applianc.simData_out is written in parallel into the database. Although the data are fed on eight nodes, the database might distribute the data on more nodes.

When a high-performance analytical procedure executes in single-machine mode, all output objects are created on the client. If the libref of the output data sets points to the appliance, the data are transferred to the database on the appliance. This can lead to considerable performance degradation compared to execution in distributed mode.

Many procedures in SAS software add the variables from the input data set when an observationwise output data set is created. The assumption of high-performance analytical procedures is that the input data sets can be large and contain many variables. For performance reasons, the output data set contains the following:

- variables that are explicitly created by the statement
- variables that are listed in the ID statement
- distribution keys or hash keys that are transferred from the input data set

Including this information enables you to add to the output data set information necessary for subsequent SQL joins without copying the entire input data set to the output data set.

Working with Formats

You can use SAS formats and user-defined formats with high-performance analytical procedures as you can with other procedures in the SAS System. However, because the analytic work is carried out in a distributed environment and might depend on the formatted values of variables, some special handling can improve the efficiency of work with formats.

High-performance analytical procedures examine the variables that are used in an analysis for association with user-defined formats. Any user-defined formats that are found by a procedure are transmitted automatically to the appliance. If you are running multiple high-performance analytical procedures in a SAS session and the analysis variables depend on user-defined formats, you can preprocess the formats. This step involves generating an XML stream (a file) of the formats and passing the stream to the high-performance analytical procedures.

Suppose that the following formats are defined in your SAS program:

```
proc format;
     value YesNo        1='Yes'        0='No';
     value checkThis    1='ThisisOne'  2='ThisisTwo';
     value $cityChar    1='Portage'    2='Kinston';
run;
```

The next group of SAS statements create the XML stream for the formats in the file *Myfmt.xml*, associate that file with the file reference myxml, and pass the file reference with the FMTLIBXML= option in the PROC HPLOGISTIC statement:

```
filename myxml 'Myfmt.xml';
libname  myxml XML92 xmltype=sasfmt tagset=tagsets.XMLsuv;
proc format cntlout=myxml.allfmts;
run;


proc hplogistic data=six fmtlibxml=myxml;
   class wheeze cit age;
   format wheeze best4. cit $cityChar.;
   model wheeze = cit age;
run;
```

Generation and destruction of the stream can be wrapped in convenience macros:

```
%macro Make_XMLStream(name=tempxml);
    filename &name 'fmt.xml';
    libname  &name XML92 xmltype=sasfmt tagset=tagsets.XMLsuv;
    proc format cntlout=&name..allfmts;
    run;
%mend;

%macro Delete_XMLStream(fref);
    %let rc=%sysfunc(fdelete(&fref));
%mend;
```

If you do not pass an XML stream to a high-performance analytical procedure that supports the FMTLIBXML= option, the procedure generates an XML stream as needed when it is invoked.

PERFORMANCE Statement

PERFORMANCE < *performance-options* > **;**

The PERFORMANCE statement defines performance parameters for multithreaded and distributed computing, passes variables that describe the distributed computing environment, and requests detailed results about the performance characteristics of a high-performance analytical procedure.

You can also use the PERFORMANCE statement to control whether a high-performance analytical procedure executes in single-machine or distributed mode.

You can specify the following *performance-options* in the PERFORMANCE statement:

COMMIT=*n*

requests that the high-performance analytical procedure write periodic updates to the SAS log when observations are sent from the client to the appliance for distributed processing.

High-performance analytical procedures do not have to use input data that are stored on the appliance. You can perform distributed computations regardless of the origin or format of the input data, provided that the data are in a format that can be read by the SAS System (for example, because a SAS/ACCESS engine is available).

In the following example, the HPREG procedure performs LASSO variable selection where the input data set is stored on the client:

```
proc hpreg data=work.one;
   model y = x1-x500;
   selection method=lasso;
   performance nodes=10 host='mydca' commit=10000;
run;
```

In order to perform the work as requested using 10 nodes on the appliance, the data set Work.One needs to be distributed to the appliance.

High-performance analytical procedures send the data in blocks to the appliance. Whenever the number of observations sent exceeds an integer multiple of the COMMIT= size, a SAS log message is produced. The message indicates the actual number of observations distributed, and not an integer multiple of the COMMIT= size.

DATASERVER=*"name"*

specifies the name of the server on Teradata systems as defined through the *hosts* file and as used in the LIBNAME statement for Teradata. For example, assume that the *hosts* file defines the server for Teradata as follows:

```
myservercop1  33.44.55.66
```

Then a LIBNAME specification would be as follows:

```
libname TDLib teradata server=myserver user= password= database= ;
```

A PERFORMANCE statement to induce running alongside the Teradata server would specify the following:

```
performance dataserver="myserver";
```

The DATASERVER= option is not required if you specify the GRIDMODE=option in the PERFORMANCE statement or if you set the GRIDMODE environment variable.

Specifying the DATASERVER= option overrides the GRIDDATASERVER environment variable.

DETAILS

requests a table that shows a timing breakdown of the procedure steps.

GRIDHOST=*"name"*

HOST=*"name"*

specifies the name of the appliance host in single or double quotation marks. If this option is specified, it overrides the value of the GRIDHOST environment variable.

GRIDMODE=SYM | ASYM

MODE=SYM | ASYM

specifies whether the high-performance analytical procedure runs in symmetric (SYM) mode or asymmetric (ASYM) mode. The default is GRIDMODE=SYM. For more information about these modes, see the section "Symmetric and Asymmetric Distributed Modes" on page 7.

If this option is specified, it overrides the GRIDMODE environment variable.

GRIDTIMEOUT=*s*

TIMEOUT=*s*

specifies the time-out in seconds for a high-performance analytical procedure to wait for a connection to the appliance and establish a connection back to the client. The default is 120 seconds. If jobs are submitted to the appliance through workload management tools that might suspend access to the appliance for a longer period, you might want to increase the time-out value.

INSTALL=*"name"*

INSTALLLOC=*"name"*

specifies the directory in which the shared libraries for the high-performance analytical procedure are installed on the appliance. Specifying the INSTALL= option overrides the GRIDINSTALLLOC environment variable.

LASRSERVER=*"path"*

LASR=*"path"*

specifies the fully qualified path to the description file of a SAS LASR Analytic Server instance. If the input data set is held in memory by this LASR Analytic Server instance, then the procedure runs alongside LASR. This option is not needed to run alongside LASR if the DATA= specification of the input data uses a libref that is associated with a LASR Analytic Server instance. For more information, see the section "Alongside-LASR Distributed Execution" on page 16 and the *SAS LASR Analytic Server: Administration Guide*.

NODES=ALL | *n*

NNODES=ALL | *n*

specifies the number of nodes in the distributed computing environment, provided that the data are not processed alongside the database.

Specifying NODES=0 indicates that you want to process the data in single-machine mode on the client machine. If the input data are not alongside the database, this is the default. The high-performance analytical procedures then perform the analysis on the client. For example, the following sets of statements are equivalent:

```
proc hplogistic data=one;
   model y = x;
run;
```

```
proc hplogistic data=one;
   model y = x;
   performance nodes=0;
run;
```

If the data are not read alongside the database, the NODES= option specifies the number of nodes on the appliance that are involved in the analysis. For example, the following statements perform the analysis in distributed mode by using 10 units of work on the appliance that is identified in the HOST= option:

```
proc hplogistic data=one;
   model y = x;
   performance nodes=10 host="hpa.sas.com";
run;
```

If the number of nodes can be modified by the application, you can specify a NODES=*n* option, where *n* exceeds the number of physical nodes on the appliance. The SAS High-Performance Econometrics software then *oversubscribes* the nodes and associates nodes with multiple units of work. For example, on a system that has 16 appliance nodes, the following statements oversubscribe the system by a factor of 3:

```
proc hplogistic data=one;
   model y = x;
   performance nodes=48 host="hpa.sas.com";
run;
```

Usually, it is not advisable to oversubscribe the system because the analytic code is optimized for a certain level of multithreading on the nodes that depends on the CPU count. You can specify NODES=ALL if you want to use all available nodes on the appliance without oversubscribing the system.

If the data are read alongside the distributed database on the appliance, specifying a nonzero value for the NODES= option has no effect. The number of units of work in the distributed computing environment is then determined by the distribution of the data and cannot be altered. For example, if you are running alongside an appliance with 24 nodes, the NODES= option in the following statements is ignored:

```
libname GPLib greenplm server=gpdca user=XXX password=YYY
              database=ZZZ;
proc hplogistic data=gplib.one;
   model y = x;
   performance nodes=10 host="hpa.sas.com";
run;
```

NTHREADS=*n*

THREADS=*n*

specifies the number of threads for analytic computations and overrides the SAS system option THREADS | NOTHREADS. If you do not specify the NTHREADS= option, the number of threads is determined based on the number of CPUs on the host on which the analytic computations execute. The algorithm by which a CPU count is converted to a thread count is specific to the high-performance analytical procedure. Most procedures create one thread per CPU for the analytic computations.

By default, high-performance analytical procedures execute in multiple concurrent threads unless multithreading has been turned off by the NOTHREADS system option or you force single-threaded execution by specifying NTHREADS=1. The largest number that can be specified for *n* is 256. Individual high-performance analytical procedures can impose more stringent limits if called for by algorithmic considerations.

NOTE: The SAS system options THREADS | NOTHREADS apply to the client machine on which the SAS high-performance analytical procedures execute. They do not apply to the compute nodes in a distributed environment.

Chapter 3
The HPCOUNTREG Procedure

Contents

Overview: HPCOUNTREG Procedure

The HPCOUNTREG procedure is a high-performance version of the COUNTREG procedure in SAS/ETS software. Like the COUNTREG procedure, the HPCOUNTREG procedure fits regression models in which the dependent variable takes on nonnegative integer or count values. Unlike the COUNTREG procedure, which can be run only on an individual workstation, the HPCOUNTREG procedure takes advantage of a computing environment that enables it to distribute the optimization task among one or more nodes. In addition, each node can use one or more threads to carry out the optimization on its subset of the data. When several nodes are employed, with each node using several threads to carry out its part of the work, the result is a highly parallel computation that provides a dramatic gain in performance.

The HPCOUNTREG procedure enables you to read and write data in distributed form and perform analyses in distributed mode and single-machine mode. For information about how to affect the execution mode of SAS high-performance analytical procedures, see the section "Processing Modes" on page 6 in Chapter 2, "Shared Concepts and Topics."

The HPCOUNTREG procedure is specifically designed to operate in the high-performance distributed environment. By default, PROC HPCOUNTREG performs computations in multiple threads.

PROC HPCOUNTREG Features

The HPCOUNTREG procedure estimates the parameters of a count regression model by maximum likelihood techniques. The following list summarizes some basic features of the HPCOUNTREG procedure:

- can perform analysis on a massively parallel high-performance appliance
- reads input data in parallel and writes output data in parallel when the data source is the appliance database
- is highly multithreaded during all phases of analytic execution
- performs maximum likelihood estimation
- supports multiple link functions

The HPCOUNTREG procedure does not support the CLASS statement.

Getting Started: HPCOUNTREG Procedure

Except for its ability to operate in the high-performance distributed environment, the HPCOUNTREG procedure is similar in use to other regression model procedures in the SAS System. For example, the following statements are used to estimate a Poisson regression model:

```
proc hpcountreg data=one ;
   model y = x / dist=poisson ;
run;
```

The response variable y is numeric and has nonnegative integer values.

This section illustrates two simple examples that use PROC HPCOUNTREG. The data are taken from Long (1997). This study examines how factors such as gender (fem), marital status (mar), number of young children (kid5), prestige of the graduate program (phd), and number of articles published by a scientist's mentor (ment) affect the number of articles (art) published by the scientist.

The first 10 observations are shown in Figure 3.1.

Figure 3.1 Article Count Data

```
Obs    art    fem    mar    kid5      phd         ment

  1     3      0      1      2     1.38000      8.0000
  2     0      0      0      0     4.29000      7.0000
  3     4      0      0      0     3.85000     47.0000
  4     1      0      1      1     3.59000     19.0000
  5     1      0      1      0     1.81000      0.0000
  6     1      0      1      1     3.59000      6.0000
  7     0      0      1      1     2.12000     10.0000
  8     0      0      1      0     4.29000      2.0000
  9     3      0      1      2     2.58000      2.0000
 10     3      0      1      1     1.80000      4.0000
```

The following SAS statements estimate the Poisson regression model. The model is executed in the distributed computing environment with two threads and four nodes.

```
/*-- Poisson Regression --*/
proc hpcountreg data=long97data;
   model art = fem mar kid5 phd ment / dist=poisson method=quanew;
   performance nthreads=2 nodes=4 details;
run;
```

The "Model Fit Summary" table that is shown in Figure 3.2 lists several details about the model. By default, the HPCOUNTREG procedure uses the Newton-Raphson optimization technique. The maximum log-likelihood value is shown, in addition to two information measures—Akaike's information criterion (AIC) and Schwarz's Bayesian information criterion (SBC)—which can be used to compare competing Poisson models. Smaller values of these criteria indicate better models.

Figure 3.2 Estimation Summary Table for a Poisson Regression

```
                         The HPCOUNTREG Procedure

                            Model Fit Summary

                  Dependent Variable                          art
                  Number of Observations                      915
                  Data Set                         WORK.LONG97DATA
                  Model                                   Poisson
                  Log Likelihood                            -1651
                  Maximum Absolute Gradient             0.0002080
                  Number of Iterations                         13
                  Optimization Method                 Quasi-Newton
                  AIC                                        3314
                  SBC                                        3343
```

Figure 3.3 shows the parameter estimates of the model and their standard errors. All covariates are significant predictors of the number of articles, except for the prestige of the program (phd), which has a p-value of 0.6271.

Figure 3.3 Parameter Estimates of Poisson Regression

```
                           Parameter Estimates

                                             Standard
       Parameter       DF     Estimate          Error    t Value    Pr > |t|

       Intercept        1       0.3046         0.1030       2.96      0.0031
       fem              1      -0.2246        0.05461      -4.11      <.0001
       mar              1       0.1552        0.06137       2.53      0.0114
       kid5             1      -0.1849        0.04013      -4.61      <.0001
       phd              1      0.01282        0.02640       0.49      0.6271
       ment             1      0.02554       0.002006      12.73      <.0001
```

To allow for variance greater than the mean, you can fit the negative binomial model instead of the Poisson model by specifying the DIST=NEGBIN option, as shown in the following statements. Whereas the Poisson model requires that the conditional mean and conditional variance be equal, the negative binomial model allows for overdispersion, in which the conditional variance can exceed the conditional mean.

```
/*-- Negative Binomial Regression --*/
proc hpcountreg data=long97data;
   model art = fem mar kid5 phd ment / dist=negbin(p=2) method=quanew;
   performance nthreads=2 nodes=4 details;
run;
```

Figure 3.4 shows the fit summary and Figure 3.5 shows the parameter estimates.

Figure 3.4 Estimation Summary Table for a Negative Binomial Regression

```
                    The HPCOUNTREG Procedure

                       Model Fit Summary

          Dependent Variable                          art
          Number of Observations                      915
          Data Set                          WORK.LONG97DATA
          Model                                    NegBin
          Log Likelihood                            -1561
          Maximum Absolute Gradient             0.0000666
          Number of Iterations                         16
          Optimization Method                 Quasi-Newton
          AIC                                        3136
          SBC                                        3170
```

Figure 3.5 Parameter Estimates of Negative Binomial Regression

```
                        Parameter Estimates

                                        Standard
   Parameter       DF     Estimate        Error     t Value     Pr > |t|

   Intercept        1       0.2561       0.1386        1.85       0.0645
   fem              1      -0.2164      0.07267       -2.98       0.0029
   mar              1       0.1505      0.08211        1.83       0.0668
   kid5             1      -0.1764      0.05306       -3.32       0.0009
   phd              1      0.01527      0.03604        0.42       0.6718
   ment             1      0.02908     0.003470        8.38       <.0001
   _Alpha           1       0.4416      0.05297        8.34       <.0001
```

The parameter estimate for _Alpha of 0.4416 is an estimate of the dispersion parameter in the negative binomial distribution. A t test for the hypothesis $H_0 : \alpha = 0$ is provided. It is highly significant, indicating overdispersion ($p < 0.0001$).

The null hypothesis $H_0 : \alpha = 0$ can be also tested against the alternative $\alpha > 0$ by using the likelihood ratio test, as described by Cameron and Trivedi (1998, pp. 45, 77–78). The likelihood ratio test statistic is equal to $-2(\mathcal{L}_P - \mathcal{L}_{NB}) = -2(-1651 + 1561) = 180$, which is highly significant, providing strong evidence of overdispersion.

Syntax: HPCOUNTREG Procedure

The following statements are available in the HPCOUNTREG procedure. Items within angle brackets (< >) or square brackets ([ ]) are optional.

PROC HPCOUNTREG *<options>* ;
  BOUNDS *bound1* [, *bound2 . . .*] ;
  FREQ *freq-variable* ;
  INIT *initialization1* < , *initialization2 . . .* > ;
  MODEL *dependent-variable* = *regressors* </ *options*> ;
  OUTPUT *<output-options>* ;
  PERFORMANCE *performance-options* ;
  RESTRICT *restriction1* [, *restriction2 . . .*] ;
  WEIGHT *variable* </ *option*> ;
  ZEROMODEL *dependent-variable* ∼ *zero-inflated-regressors* </ *options*> ;

There can be only one MODEL statement. The ZEROMODEL statement, if used, must appear after the MODEL statement. If a FREQ or WEIGHT statement is specified more than once, the variable specified in the first instance is used.

Functional Summary

Table 3.1 summarizes the statements and options used with the HPCOUNTREG procedure.

Table 3.1 HPCOUNTREG Functional Summary

Description	Statement	Option
Data Set Options		
Specifies the input data set	HPCOUNTREG	DATA=
Writes parameter estimates to an output data set	HPCOUNTREG	OUTEST=
Writes estimates to an output data set	OUTPUT	OUT=
Specifies an optional frequency variable	FREQ	
Specifies an optional weight variable	WEIGHT	
Printing Control Options		
Prints the correlation matrix of the estimates	HPCOUNTREG	CORRB
Prints the covariance matrix of the estimates	HPCOUNTREG	COVB
Suppresses the normal printed output	HPCOUNTREG	NOPRINT
Requests all printing options	HPCOUNTREG	PRINTALL
Options to Control the Optimization Process		
Specifies maximum number of iterations allowed	HPCOUNTREG	MAXITER=
Selects the iterative minimization method to use	HPCOUNTREG	METHOD=
Specifies maximum number of iterations allowed	HPCOUNTREG	MAXITER=
Specifies maximum number of function calls	HPCOUNTREG	MAXFUNC=
Specifies the upper limit of CPU time in seconds	HPCOUNTREG	MAXTIME=
Specifies absolute function convergence criterion	HPCOUNTREG	ABSCONV=
Specifies absolute function convergence criterion	HPCOUNTREG	ABSFCONV=
Specifies absolute gradient convergence criterion	HPCOUNTREG	ABSGCONV=
Specifies relative function convergence criterion	HPCOUNTREG	FCONV=
Specifies relative gradient convergence criterion	HPCOUNTREG	GCONV=

Description	Statement	Option
Specifies absolute parameter convergence criterion	HPCOUNTREG	ABSXCONV=
Specifies matrix singularity criterion	HPCOUNTREG	SINGULAR=
Sets boundary restrictions on parameters	BOUNDS	
Sets initial values for parameters	INIT	
Sets linear restrictions on parameters	RESTRICT	
Model Estimation Options		
Specifies the type of model	HPCOUNTREG	DIST=
Specifies the type of covariance matrix	HPCOUNTREG	COVEST=
Suppresses the intercept parameter	MODEL	NOINT
Specifies the offset variable	MODEL	OFFSET=
Specifies the zero-inflated offset variable	ZEROMODEL	OFFSET=
Specifies the zero-inflated link function	ZEROMODEL	LINK=
Output Control Options		
Includes covariances in the OUTEST= data set	HPCOUNTREG	COVOUT
Includes correlations in the OUTEST= data set	HPCOUNTREG	CORROUT
Outputs SAS variables to the output data set	OUTPUT	COPYVAR=
Outputs probability of the actual value	OUTPUT	PROB=
Outputs expected value of response variable	OUTPUT	PRED=
Outputs estimates of XBeta $= \mathbf{x}_i' \boldsymbol{\beta}$	OUTPUT	XBETA=
Outputs estimates of ZGamma $= \mathbf{z}_i' \boldsymbol{\gamma}$	OUTPUT	ZGAMMA=
Outputs probability of a zero value as a result of the zero-generating process	OUTPUT	PROBZERO=
Performance Options		
Requests a table that shows a timing breakdown	PERFORMANCE	DETAILS
Specifies the number of threads to use	PERFORMANCE	NTHREADS=
Specifies the number of nodes to use on the SAS appliance	PERFORMANCE	NODES=

PROC HPCOUNTREG Statement

PROC HPCOUNTREG <*options*> **;**

The following *options* can be used in the PROC HPCOUNTREG statement.

Input Data Set Option

DATA=*SAS-data-set*

specifies the input SAS data set. If the DATA= option is not specified, PROC HPCOUNTREG uses the most recently created SAS data set.

Output Data Set Options

OUTEST=*SAS-data-set*
writes the parameter estimates to the specified output data set.

CORROUT
writes the correlation matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

COVOUT
writes the covariance matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

Printing Options

You can specify the following options in either the PROC HPCOUNTREG statement or the MODEL statement:

CORRB
prints the correlation matrix of the parameter estimates.

COVB
prints the covariance matrix of the parameter estimates.

NOPRINT
suppresses all printed output.

PRINTALL
requests all printing options.

Estimation Control Options

You can specify the following options in either the PROC HPCOUNTREG statement or the MODEL statement:

COVEST=*value*
specifies the type of covariance matrix for the parameter estimates.

The default is COVEST=HESSIAN. You can specify the following *values*:

HESSIAN	specifies the covariance from the Hessian matrix.
OP	specifies the covariance from the outer product matrix.
QML	specifies the covariance from the outer product and Hessian matrices.

Optimization Control Options

PROC HPCOUNTREG uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks. You can specify the following *options* in either the PROC HPCOUNTREG statement or the MODEL statement.

ABSCONV=*r*

ABSTOL=*r*

specifies an absolute function value convergence criterion by which minimization stops when $f(\theta^{(k)}) \leq r$. The default value of *r* is the negative square root of the largest double-precision value, which serves only as a protection against overflows.

ABSFCONV=*r*

ABSFTOL=*r*

specifies an absolute function difference convergence criterion by which minimization stops when the function value has a small change in successive iterations:

$$|f(\theta^{(k-1)}) - f(\theta^{(k)})| \leq r$$

The default is 0.

ABSGCONV=*r*

ABSGTOL=*r*

specifies an absolute gradient convergence criterion. Optimization stops when the maximum absolute gradient element is small:

$$\max_j |g_j(\theta^{(k)})| \leq r$$

The default is 1E–5.

ABSXCONV=*r*

ABSXTOL=*r*

specifies an absolute parameter convergence criterion. Optimization stops when the Euclidean distance between successive parameter vectors is small:

$$\parallel \theta^{(k)} - \theta^{(k-1)} \parallel_2 \leq r$$

The default is 0.

FCONV=*r*

FTOL=*r*

specifies a relative function convergence criterion. Optimization stops when a relative change of the function value in successive iterations is small:

$$\frac{|f(\theta^{(k)}) - f(\theta^{(k-1)})|}{|f(\theta^{(k-1)})|} \leq r$$

The default value is 2ϵ, where ϵ denotes the machine precision constant, which is the smallest double-precision floating-point number such that $1 + \epsilon > 1$.

GCONV=*r*

GTOL=*r*

specifies a relative gradient convergence criterion. For all techniques except CONGRA, optimization stops when the normalized predicted function reduction is small:

$$\frac{g(\theta^{(k)})^T [H^{(k)}]^{-1} g(\theta^{(k)})}{|f(\theta^{(k)})|} \leq r$$

For the CONGRA technique (where a reliable Hessian estimate H is not available), the following criterion is used:

$$\frac{\| g(\theta^{(k)}) \|_2^2 \quad \| s(\theta^{(k)}) \|_2}{\| g(\theta^{(k)}) - g(\theta^{(k-1)}) \|_2 \, |f(\theta^{(k)})|} \leq r$$

The default is 1E–8.

MAXFUNC=*i*

MAXFU=*i*

specifies the maximum number of function calls in the optimization process. The default is 1,000.

The optimization can terminate only after completing a full iteration. Therefore, the number of function calls that are actually performed can exceed the number of calls that are specified by this option.

MAXITER=*i*

MAXIT=*i*

specifies the maximum number of iterations in the optimization process. The default is 200.

MAXTIME=*r*

specifies an upper limit of *r* seconds of CPU time for the optimization process. The default value is the largest floating-point double representation of your computer. The time that is specified by this option is checked only once at the end of each iteration. Therefore, the actual run time can be much longer than *r*. The actual run time includes the remaining time needed to finish the iteration and the time needed to generate the output of the results.

METHOD=*value*

specifies the iterative minimization method to use. The default is METHOD=NEWRAP. You can specify the following *values*:

CONGRA	specifies the conjugate-gradient method.
DBLDOG	specifies the double-dogleg method.
NEWRAP	specifies the Newton-Raphson method (this is the default).
NONE	specifies that no optimization be performed beyond using the ordinary least squares method to compute the parameter estimates.
NRRIDG	specifies the Newton-Raphson Ridge method.
QUANEW	specifies the quasi-Newton method.
TRUREG	specifies the trust region method.

SINGULAR=*r*

specifies the general singularity criterion that is applied by the HPCOUNTREG procedure in sweeps and inversions. The default is 1E–8.

BOUNDS Statement

BOUNDS *bound1* [*, bound2 . . .*] **;**

The BOUNDS statement imposes simple boundary constraints on the parameter estimates. You can specify any number of BOUNDS statements.

Each *bound* is composed of parameter names, constants, and inequality operators as follows:

item operator item [*operator item* [*operator item . . .*]]

Each *item* is a constant, a parameter name, or a list of parameter names. Each *operator* is <, >, <=, or >=. Parameter names are as shown in the Effect column of the "Parameter Estimates" table.

You can use both the BOUNDS statement and the RESTRICT statement to impose boundary constraints. However, the BOUNDS statement provides a simpler syntax for specifying these kinds of constraints. For more information, see the section "RESTRICT Statement" on page 52.

The following BOUNDS statement illustrates the use of parameter lists to specify boundary constraints. It constrains the estimates of the parameter for z to be negative, the parameters for x1 through x10 to be between 0 and 1, and the parameter for x1 in the zero-inflation model to be less than 1.

```
bounds z < 0,
       0 < x1-x10 < 1,
       Inf_x1 < 1;
```

FREQ Statement

FREQ *freq-variable* **;**

The FREQ statement identifies a variable (*freq-variable*) that contains the frequency of occurrence of each observation. PROC HPCOUNTREG treats each observation as if it appears n times, where n is the value of *freq-variable* for the observation. If the value for the observation is not an integer, it is truncated to an integer. If the value is less than 1 or missing, the observation is not used in the model fitting. When the FREQ statement is not specified, each observation is assigned a frequency of 1.

INIT Statement

INIT *initialization1* < *, initialization2 . . .*> **;**

The INIT statement sets initial values for parameters in the optimization.

Each *initialization* is written as a parameter or parameter list, followed by an optional equal sign (=), followed by a number:

parameter <=> *number*

Parameter names are as shown in the Effect column of the "Parameter Estimates" table.

MODEL Statement

MODEL *dependent-variable* **=** *regressors* *</ options>* **;**

The MODEL statement specifies the dependent variable and independent regressor variables for the regression model. The dependent count variable should take only nonnegative integer values from the input data set. PROC HPCOUNTREG rounds any positive noninteger count value to the nearest integer. PROC HPCOUNTREG discards any observation that has a negative count.

Only one MODEL statement can be specified. You can specify the following *options* in the MODEL statement after a slash (/).

DIST=*value*

specifies a type of model to be analyzed. You can specify the following *values*:

POISSON | P	specifies the Poisson regression model.
NEGBIN(P=1)	specifies the negative binomial regression model that uses a linear variance function.
NEGBIN(P=2) | NEGBIN	specifies the negative binomial regression model that uses a quadratic variance function.
ZIPOISSON | ZIP	specifies zero-inflated Poisson regression.
ZINEGBIN | ZINB	specifies zero-inflated negative binomial regression.

You can also specify the DIST option in the HPCOUNTREG statement.

NOINT

suppresses the intercept parameter.

OFFSET=*offset-variable*

specifies a variable in the input data set to be used as an offset variable. The *offset-variable* is used to allow the observational units to vary across observations. For example, when the number of shipping accidents could be measured across different time periods or the number of students who participate in an activity could be reported across different class sizes, the observational units need to be adjusted to a common denominator by using the offset variable. The offset variable appears as a covariate in the model with its parameter restricted to 1. The offset variable cannot be the response variable, the zero-inflation offset variable (if any), or any of the explanatory variables. The "Model Fit Summary" table gives the name of the data set variable that is used as the offset variable; it is labeled "Offset."

Printing Options

You can specify the following options in either the PROC HPCOUNTREG statement or the MODEL statement:

CORRB

prints the correlation matrix of the parameter estimates.

COVB

prints the covariance matrix of the parameter estimates.

NOPRINT
suppresses all printed output.

PRINTALL
requests all printing options.

OUTPUT Statement

OUTPUT < *output-options* > **;**

The OUTPUT statement creates a new SAS data set that includes variables created by the *output-options*. These variables include the estimates of $\mathbf{x}_i'\boldsymbol{\beta}$, the expected value of the response variable, and the probability of the response variable taking on the current value. Furthermore, if a zero-inflated model was fit, you can request that the output data set contain the estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$ and the probability that the response is zero as a result of the zero-generating process. These statistics can be computed for all observations in which the regressors are not missing, even if the response is missing. By adding observations that have missing response values to the input data set, you can compute these statistics for new observations or for settings of the regressors that are not present in the data without affecting the model fit.

You can specify only one OUTPUT statement. You can specify the following *output-options*:

OUT=*SAS-data-set*
names the output data set

COPYVAR=*SAS-variable-names*
COPYVARS=*SAS-variable-names*
adds SAS variables to the output data set.

PRED=*name*
names the variable to contain the predicted value of the response variable.

PROB=*name*
names the variable to contain the probability that the response variable will take the actual value, $\Pr(Y = y_i)$.

PROBCOUNT(*value1* **<** *value2* **. . . >)**
outputs the probability that the response variable will take particular values. Each value should be a nonnegative integer. Nonintegers are rounded to the nearest integer. For *value*, you can also specify a list of the form X TO Y BY Z. For example, PROBCOUNT(0 1 2 TO 10 BY 2 15) requests predicted probabilities for the counts 0, 1, 2, 4, 5, 6, 8, 10, and 15. This option is not available for the fixed- and random-effects panel models.

PROBZERO=*name*
names the variable to contain the value of φ_i, which is the probability that the response variable will take the value of 0 as a result of the zero-generating process. This variable is written to the output file only if the model is zero-inflated.

XBETA=*name*
names the variable to contain estimates of $\mathbf{x}_i'\boldsymbol{\beta}$.

ZGAMMA=*name*
names the variable to contain estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$.

PERFORMANCE Statement

PERFORMANCE < *performance-options* > **;**

The PERFORMANCE statement specifies options to control the multithreaded and distributed computing environment and requests detailed results about the performance characteristics of the HPCOUNTREG procedure. You can also use the PERFORMANCE statement to control whether the HPCOUNTREG procedure executes in single-machine or distributed mode. The most commonly used *performance-options* in the PERFORMANCE statement are as follows:

DETAILS
requests a table that shows a timing breakdown of the procedure steps.

NODES=*n*
specifies the number of nodes in the distributed computing environment, provided that the data are not processed alongside the database.

NTHREADS=*n*
specifies the number of threads for analytic computations and overrides the SAS system option THREADS | NOTHREADS. If you do not specify the NTHREADS= option, PROC HPCOUNTREG creates one thread per CPU for the analytic computations.

For more information about the PERFORMANCE statement for high-performance analytical procedures, see the section "PERFORMANCE Statement" on page 34 of Chapter 2, "Shared Concepts and Topics."

RESTRICT Statement

RESTRICT *restriction1* [, *restriction2* . . .] **;**

The RESTRICT statement imposes linear restrictions on the parameter estimates. You can specify any number of RESTRICT statements.

Each *restriction* is written as an expression, followed by an equality operator (=) or an inequality operator (<, >, <=, >=) and then by a second expression, as follows:

expression operator expression

The *operator* can be =, <, >, <=, or >=.

Restriction expressions can be composed of parameter names, constants, and the following operators: times ($*$), plus ($+$), and minus ($-$). Parameter names are as shown in the Effect column of the "Parameter Estimates" table. The restriction expressions must be a linear function of the variables.

Lagrange multipliers are reported in the "Parameter Estimates" table for all the active linear constraints. They are identified by the names Restrict1, Restrict2, and so on. The probabilities of these Lagrange multipliers are computed using a beta distribution (LaMotte 1994). Nonactive (nonbinding) restrictions have no effect on the estimation results and are not noted in the output.

The following RESTRICT statement constrains the negative binomial dispersion parameter α to 1, which restricts the conditional variance to be $\mu + \mu^2$:

```
restrict _Alpha = 1;
```

WEIGHT Statement

WEIGHT *variable* < / *option*> **;**

The WEIGHT statement specifies a variable to supply weighting values to use for each observation in estimating parameters. The log likelihood for each observation is multiplied by the corresponding weight variable value.

If the weight of an observation is nonpositive, that observation is not used in the estimation.

The following *option* can be added to the WEIGHT statement after a slash (/).

NONORMALIZE

does not normalize the weights. (By default, the weights are normalized so that they add up to the actual sample size. The weights w_i are normalized by multiplying them by $\frac{n}{\sum_{i=1}^{n} w_i}$, where n is the sample size.) If the weights are required to be used as they are, then specify the NONORMALIZE option.

ZEROMODEL Statement

ZEROMODEL *dependent-variable* $\sim$ *zero-inflated-regressors* < / *options* > **;**

The ZEROMODEL statement is required if either ZIP or ZINB is specified in the DIST= option in the MODEL statement. If ZIP or ZINB is specified, then the ZEROMODEL statement must follow the MODEL statement. The dependent variable in the ZEROMODEL statement must be the same as the dependent variable in the MODEL statement.

The zero-inflated (ZI) regressors appear in the equation that determines the probability (φ_i) of a zero count. Each of these q variables has a parameter to be estimated in the regression. For example, let $\mathbf{z}_i'$ be the ith observation's $1 \times (q + 1)$ vector of values of the q ZI explanatory variables (w_0 is set to 1 for the intercept term). Then φ_i is a function of $\mathbf{z}_i' \boldsymbol{\gamma}$, where $\boldsymbol{\gamma}$ is the $(q + 1) \times 1$ vector of parameters to be estimated. (The zero-inflated intercept is γ_0; the coefficients for the q zero-inflated covariates are $\gamma_1, \ldots, \gamma_q$.) If q is equal to 0 (no ZI explanatory variables are provided), then only the intercept term γ_0 is estimated. The "Parameter Estimates" table in the displayed output shows the estimates for the ZI intercept and ZI explanatory variables; they are labeled with the prefix "Inf_". For example, the ZI intercept is labeled "Inf_intercept". If you specify Age (a variable in your data set) as a ZI explanatory variable, then the "Parameter Estimates" table labels the corresponding parameter estimate "Inf_Age".

You can specify the following *options* in the ZEROMODEL statement after a slash (/):

LINK=*value*

specifies the distribution function used to compute probability of zeros. The supported distribution functions are as follows:

LOGISTIC	specifies logistic distribution.
NORMAL	specifies standard normal distribution.

If this option is omitted, then the default ZI link function is logistic.

OFFSET=*zero-inflated-offset-variable*

specifies a variable in the input data set to be used as a zero-inflated (ZI) offset variable. The ZI offset variable *zero-inflated-offset-variable* is included as a term, with coefficient restricted to 1, in the equation that determines the probability (φ_i) of a zero count and represents an adjustment to a common observational unit. The ZI offset variable cannot be the response variable, the offset variable (if any), or any of the explanatory variables. The name of the data set variable that is used as the ZI offset variable is displayed in the "Model Fit Summary" table, where it is labeled as "Inf_offset".

Details: HPCOUNTREG Procedure

Missing Values

Any observations in the input data set that have a missing value for one or more of the regressors are ignored by PROC HPCOUNTREG and not used in the model fit. PROC HPCOUNTREG rounds any positive noninteger count values to the nearest integer and ignores any observations that have a negative count.

If the input data set contains any observations that have missing response values but nonmissing regressors, PROC HPCOUNTREG can compute several statistics and store them in an output data set by using the OUTPUT statement. For example, you can request that the output data set contain the estimates of $\mathbf{x}_i'\boldsymbol{\beta}$, the expected value of the response variable, and the probability that the response variable will take the current value. Furthermore, if a zero-inflated model was fit, you can request that the output data set contain the estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$, and the probability that the response is 0 as a result of the zero-generating process. Note that the presence of such observations (that have missing response values) does not affect the model fit.

Poisson Regression

The most widely used model for count data analysis is Poisson regression. Poisson regression assumes that y_i, given the vector of covariates $\mathbf{x}_i$, is independently Poisson distributed with

$$P(Y_i = y_i|\mathbf{x}_i) = \frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!}, \quad y_i = 0, 1, 2, \ldots$$

and the mean parameter—that is, the mean number of events per period—is given by

$$\mu_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

where $\boldsymbol{\beta}$ is a $(k+1) \times 1$ parameter vector. (The intercept is β_0; the coefficients for the k regressors are $\beta_1, \ldots, \beta_k$.) Taking the exponential of $\mathbf{x}_i'\boldsymbol{\beta}$ ensures that the mean parameter μ_i is nonnegative. It can be shown that the conditional mean is given by

$$E(y_i|\mathbf{x}_i) = \mu_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

The name *log-linear model* is also used for the Poisson regression model because the logarithm of the conditional mean is linear in the parameters:

$$\ln[E(y_i|\mathbf{x}_i)] = \ln(\mu_i) = \mathbf{x}_i'\boldsymbol{\beta}$$

Note that the conditional variance of the count random variable is equal to the conditional mean in the Poisson regression model:

$$V(y_i|\mathbf{x}_i) = E(y_i|\mathbf{x}_i) = \mu_i$$

The equality of the conditional mean and variance of y_i is known as *equidispersion.*

The marginal effect of a regressor is given by

$$\frac{\partial E(y_i|\mathbf{x}_i)}{\partial x_{ji}} = \exp(\mathbf{x}_i'\boldsymbol{\beta})\boldsymbol{\beta}_j = E(y_i|\mathbf{x}_i)\beta_j$$

Thus, a one-unit change in the jth regressor leads to a proportional change in the conditional mean $E(y_i|\mathbf{x}_i)$ of β_j.

The standard estimator for the Poisson model is the maximum likelihood estimator (MLE). Because the observations are independent, the log-likelihood function is written as

$$\mathcal{L} = \sum_{i=1}^{N}(-\mu_i + y_i \ln \mu_i - \ln y_i!) = \sum_{i=1}^{N}(-e^{\mathbf{x}_i'\boldsymbol{\beta}} + y_i\mathbf{x}_i'\boldsymbol{\beta} - \ln y_i!)$$

The gradient and the Hessian, respectively, are as follows:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N}(y_i - \mu_i)\mathbf{x}_i = \sum_{i=1}^{N}(y_i - e^{\mathbf{x}_i'\boldsymbol{\beta}})\mathbf{x}_i$$

$$\frac{\partial^2 \mathcal{L}}{\partial \boldsymbol{\beta}\partial \boldsymbol{\beta}'} = -\sum_{i=1}^{N}\mu_i\mathbf{x}_i\mathbf{x}_i{}' = -\sum_{i=1}^{N}e^{\mathbf{x}_i'\boldsymbol{\beta}}\mathbf{x}_i\mathbf{x}_i'$$

The Poisson model has been criticized for its restrictive property that the conditional variance equals the conditional mean. Real-life data are often characterized by *overdispersion*—that is, the variance exceeds the mean. Allowing for overdispersion can improve model predictions because the Poisson restriction of equal mean and variance results in the underprediction of zeros when overdispersion exists. The most commonly used model that accounts for overdispersion is the negative binomial model.

Negative Binomial Regression

The Poisson regression model can be generalized by introducing an unobserved heterogeneity term for observation i. Thus, the individuals are assumed to differ randomly in a manner that is not fully accounted for by the observed covariates. This is formulated as

$$E(y_i|\mathbf{x}_i, \tau_i) = \mu_i \tau_i = e^{\mathbf{x}_i' \boldsymbol{\beta} + \epsilon_i}$$

where the unobserved heterogeneity term $\tau_i = e^{\epsilon_i}$ is independent of the vector of regressors $\mathbf{x}_i$. Then the distribution of y_i conditional on $\mathbf{x}_i$ and τ_i is Poisson with conditional mean and conditional variance $\mu_i \tau_i$:

$$f(y_i|\mathbf{x}_i, \tau_i) = \frac{\exp(-\mu_i \tau_i)(\mu_i \tau_i)^{y_i}}{y_i!}$$

Let $g(\tau_i)$ be the probability density function of τ_i. Then, the distribution $f(y_i|\mathbf{x}_i)$ (no longer conditional on τ_i) is obtained by integrating $f(y_i|\mathbf{x}_i, \tau_i)$ with respect to τ_i:

$$f(y_i|\mathbf{x}_i) = \int_0^\infty f(y_i|\mathbf{x}_i, \tau_i) g(\tau_i) d\tau_i$$

An analytical solution to this integral exists when τ_i is assumed to follow a gamma distribution. This solution is the negative binomial distribution. If the model contains a constant term, then in order to identify the mean of the distribution, it is necessary to assume that $E(e^{\epsilon_i}) = E(\tau_i) = 1$. Thus, it is assumed that τ_i follows a gamma(θ, θ) distribution with $E(\tau_i) = 1$ and $V(\tau_i) = 1/\theta$,

$$g(\tau_i) = \frac{\theta^\theta}{\Gamma(\theta)} \tau_i^{\theta-1} \exp(-\theta \tau_i)$$

where $\Gamma(x) = \int_0^\infty z^{x-1} \exp(-z) dz$ is the gamma function and θ is a positive parameter. Then, the density of y_i given $\mathbf{x}_i$ is derived as

$$\begin{aligned}
f(y_i|\mathbf{x}_i) &= \int_0^\infty f(y_i|\mathbf{x}_i, \tau_i) g(\tau_i) d\tau_i \\
&= \frac{\theta^\theta \mu_i^{y_i}}{y_i! \Gamma(\theta)} \int_0^\infty e^{-(\mu_i + \theta)\tau_i} \tau_i^{\theta + y_i - 1} d\tau_i \\
&= \frac{\theta^\theta \mu_i^{y_i} \Gamma(y_i + \theta)}{y_i! \Gamma(\theta)(\theta + \mu_i)^{\theta + y_i}} \\
&= \frac{\Gamma(y_i + \theta)}{y_i! \Gamma(\theta)} \left(\frac{\theta}{\theta + \mu_i}\right)^\theta \left(\frac{\mu_i}{\theta + \mu_i}\right)^{y_i}
\end{aligned}$$

If you make the substitution $\alpha = \frac{1}{\theta}$ ($\alpha > 0$), the negative binomial distribution can then be rewritten as

$$f(y_i|\mathbf{x}_i) = \frac{\Gamma(y_i + \alpha^{-1})}{y_i! \Gamma(\alpha^{-1})} \left(\frac{\alpha^{-1}}{\alpha^{-1} + \mu_i}\right)^{\alpha^{-1}} \left(\frac{\mu_i}{\alpha^{-1} + \mu_i}\right)^{y_i}, \quad y_i = 0, 1, 2, \ldots$$

Thus, the negative binomial distribution is derived as a gamma mixture of Poisson random variables. It has the conditional mean

$$E(y_i|\mathbf{x}_i) = \mu_i = e^{\mathbf{x}_i' \boldsymbol{\beta}}$$

and the conditional variance

$$V(y_i|\mathbf{x}_i) = \mu_i[1 + \frac{1}{\theta}\mu_i] = \mu_i[1 + \alpha\mu_i] > E(y_i|\mathbf{x}_i)$$

The conditional variance of the negative binomial distribution exceeds the conditional mean. Overdispersion results from neglected unobserved heterogeneity. The negative binomial model with variance function $V(y_i|\mathbf{x}_i) = \mu_i + \alpha\mu_i^2$, which is quadratic in the mean, is referred to as the NEGBIN2 model (Cameron and Trivedi 1986). To estimate this model, specify DIST=NEGBIN(P=2) in the MODEL statement. The Poisson distribution is a special case of the negative binomial distribution where $\alpha = 0$. A test of the Poisson distribution can be carried out by testing the hypothesis that $\alpha = \frac{1}{\theta_i} = 0$. A Wald test of this hypothesis is provided (it is the reported t statistic for the estimated α in the negative binomial model).

The log-likelihood function of the negative binomial regression model (NEGBIN2) is given by

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^{N}\Bigg\{\sum_{j=0}^{y_i-1}\ln(j+\alpha^{-1}) - \ln(y_i!) \\ &\quad -(y_i+\alpha^{-1})\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) + y_i\ln(\alpha) + y_i\mathbf{x}_i'\boldsymbol{\beta}\Bigg\}\end{aligned}$$

where use of the following fact is made if y is an integer:

$$\Gamma(y+a)/\Gamma(a) = \prod_{j=0}^{y-1}(j+a)$$

The gradient is

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} = \sum_{i=1}^{N}\frac{y_i-\mu_i}{1+\alpha\mu_i}\mathbf{x}_i$$

and

$$\frac{\partial\mathcal{L}}{\partial\alpha} = \sum_{i=1}^{N}\left\{-\alpha^{-2}\sum_{j=0}^{y_i-1}\frac{1}{(j+\alpha^{-1})} + \alpha^{-2}\ln(1+\alpha\mu_i) + \frac{y_i-\mu_i}{\alpha(1+\alpha\mu_i)}\right\}$$

Cameron and Trivedi (1986) consider a general class of negative binomial models that have mean μ_i and variance function $\mu_i + \alpha\mu_i^p$. The NEGBIN2 model, with $p = 2$, is the standard formulation of the negative binomial model. Models that have other values of p, $-\infty < p < \infty$, have the same density $f(y_i|\mathbf{x}_i)$, except that α^{-1} is replaced everywhere by $\alpha^{-1}\mu^{2-p}$. The negative binomial model NEGBIN1, which sets $p = 1$, has the variance function $V(y_i|\mathbf{x}_i) = \mu_i + \alpha\mu_i$, which is linear in the mean. To estimate this model, specify DIST=NEGBIN(P=1) in the MODEL statement.

The log-likelihood function of the NEGBIN1 regression model is given by

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^{N}\Bigg\{\sum_{j=0}^{y_i-1}\ln\left(j+\alpha^{-1}\exp(\mathbf{x}_i'\boldsymbol{\beta})\right) \\ &\quad -\ln(y_i!) - \left(y_i+\alpha^{-1}\exp(\mathbf{x}_i'\boldsymbol{\beta})\right)\ln(1+\alpha) + y_i\ln(\alpha)\Bigg\}\end{aligned}$$

The gradient is

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N} \left\{ \left(\sum_{j=0}^{y_i-1} \frac{\mu_i}{(j\alpha + \mu_i)} \right) \mathbf{x}_i - \alpha^{-1} \ln(1+\alpha)\mu_i \mathbf{x}_i \right\}$$

and

$$\frac{\partial \mathcal{L}}{\partial \alpha} = \sum_{i=1}^{N} \left\{ - \left(\sum_{j=0}^{y_i-1} \frac{\alpha^{-1}\mu_i}{(j\alpha + \mu_i)} \right) - \alpha^{-2}\mu_i \ln(1+\alpha) - \frac{(y_i + \alpha^{-1}\mu_i)}{1+\alpha} + \frac{y_i}{\alpha} \right\}$$

Zero-Inflated Count Regression Overview

The main motivation for using zero-inflated count models is that real-life data frequently display overdispersion and excess zeros. Zero-inflated count models provide a way to both model the excess zeros and allow for overdispersion. In particular, there are two possible data generation processes for each observation. The result of a Bernoulli trial is used to determine which of the two processes to use. For observation i, Process 1 is chosen with probability φ_i and Process 2 with probability $1 - \varphi_i$. Process 1 generates only zero counts. Process 2 generates counts from either a Poisson or a negative binomial model. In general,

$$y_i \sim \begin{cases} 0 & \text{with probability} \quad \varphi_i \\ g(y_i) & \text{with probability} \quad 1 - \varphi_i \end{cases}$$

Therefore, the probability of $\{Y_i = y_i\}$ can be described as

$$\begin{aligned} P(y_i = 0|\mathbf{x}_i) &= \varphi_i + (1-\varphi_i)g(0) \\ P(y_i|\mathbf{x}_i) &= (1-\varphi_i)g(y_i), \quad y_i > 0 \end{aligned}$$

where $g(y_i)$ follows either the Poisson or the negative binomial distribution.

If the probability φ_i depends on the characteristics of observation i, then φ_i is written as a function of $\mathbf{z}_i'\boldsymbol{\gamma}$, where $\mathbf{z}_i'$ is the $1 \times (q+1)$ vector of zero-inflated covariates and $\boldsymbol{\gamma}$ is the $(q+1) \times 1$ vector of zero-inflated coefficients to be estimated. (The zero-inflated intercept is γ_0; the coefficients for the q zero-inflated covariates are $\gamma_1, \ldots, \gamma_q$.) The function F that relates the product $\mathbf{z}_i'\boldsymbol{\gamma}$ (which is a scalar) to the probability φ_i is called the zero-inflated link function,

$$\varphi_i = F_i = F(\mathbf{z}_i'\boldsymbol{\gamma})$$

In the HPCOUNTREG procedure, the zero-inflated covariates are indicated in the ZEROMODEL statement. Furthermore, the zero-inflated link function F can be specified as either the logistic function,

$$F(\mathbf{z}_i'\boldsymbol{\gamma}) = \Lambda(\mathbf{z}_i'\boldsymbol{\gamma}) = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

or the standard normal cumulative distribution function (also called the probit function),

$$F(\mathbf{z}_i'\boldsymbol{\gamma}) = \Phi(\mathbf{z}_i'\boldsymbol{\gamma}) = \int_0^{\mathbf{z}_i'\boldsymbol{\gamma}} \frac{1}{\sqrt{2\pi}} \exp(-u^2/2)du$$

The zero-inflated link function is indicated by using the LINK= option in the ZEROMODEL statement. The default ZI link function is the logistic function.

Zero-Inflated Poisson Regression

In the zero-inflated Poisson (ZIP) regression model, the data generation process that is referred to earlier as Process 2 is

$$g(y_i) = \frac{\exp(-\mu_i)\mu_i^{y_i}}{y_i!}$$

where $\mu_i = e^{\mathbf{x}_i'\boldsymbol{\beta}}$. Thus the ZIP model is defined as

$$\begin{aligned} P(y_i = 0|\mathbf{x}_i, \mathbf{z}_i) &= F_i + (1 - F_i)\exp(-\mu_i) \\ P(y_i|\mathbf{x}_i, \mathbf{z}_i) &= (1 - F_i)\frac{\exp(-\mu_i)\mu_i^{y_i}}{y_i!}, \quad y_i > 0 \end{aligned}$$

The conditional expectation and conditional variance of y_i are given by

$$E(y_i|\mathbf{x}_i, \mathbf{z}_i) = \mu_i(1 - F_i)$$

$$V(y_i|\mathbf{x}_i, \mathbf{z}_i) = E(y_i|\mathbf{x}_i, \mathbf{z}_i)(1 + \mu_i F_i)$$

Note that the ZIP model (in addition to the ZINB model) exhibits overdispersion because $V(y_i|\mathbf{x}_i, \mathbf{z}_i) > E(y_i|\mathbf{x}_i, \mathbf{z}_i)$.

In general, the log-likelihood function of the ZIP model is

$$\mathcal{L} = \sum_{i=1}^{N} \ln\left[P(y_i|\mathbf{x}_i, \mathbf{z}_i)\right]$$

After a specific link function (either logistic or standard normal) for the probability φ_i is chosen, it is possible to write the exact expressions for the log-likelihood function and the gradient.

ZIP Model with Logistic Link Function

First, consider the ZIP model in which the probability φ_i is expressed by a logistic link function, namely

$$\varphi_i = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

The log-likelihood function is

$$\begin{aligned} \mathcal{L} = & \sum_{\{i:y_i=0\}} \ln\left[\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + \exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))\right] \\ & + \sum_{\{i:y_i>0\}} \left[y_i\mathbf{x}_i'\boldsymbol{\beta} - \exp(\mathbf{x}_i'\boldsymbol{\beta}) - \sum_{k=2}^{y_i}\ln(k)\right] \\ & - \sum_{i=1}^{N}\ln\left[1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})\right] \end{aligned}$$

The gradient for this model is given by

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\gamma}} = \sum_{\{i:y_i=0\}} \left[\frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + \exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))} \right] \mathbf{z}_i - \sum_{i=1}^{N} \left[\frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1+\exp(\mathbf{z}_i'\boldsymbol{\gamma})} \right] \mathbf{z}_i$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = \sum_{\{i:y_i=0\}} \left[\frac{-\exp(\mathbf{x}_i'\boldsymbol{\beta})\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))}{\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + \exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))} \right] \mathbf{x}_i + \sum_{\{i:y_i>0\}} \left[y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta}) \right] \mathbf{x}_i$$

ZIP Model with Standard Normal Link Function

Next, consider the ZIP model in which the probability φ_i is expressed by a standard normal link function: $\varphi_i = \Phi(\mathbf{z}_i'\boldsymbol{\gamma})$. The log-likelihood function is

$$\begin{aligned} \mathcal{L} &= \sum_{\{i:y_i=0\}} \ln \left\{ \Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1 - \Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right] \exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta})) \right\} \\ &+ \sum_{\{i:y_i>0\}} \left\{ \ln\left[\left(1 - \Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right)\right] - \exp(\mathbf{x}_i'\boldsymbol{\beta}) + y_i \mathbf{x}_i'\boldsymbol{\beta} - \sum_{k=2}^{y_i} \ln(k) \right\} \end{aligned}$$

The gradient for this model is given by

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\gamma}} &= \sum_{\{i:y_i=0\}} \frac{\varphi(\mathbf{z}_i'\boldsymbol{\gamma})\left[1 - \exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))\right]}{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1 - \Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))} \mathbf{z}_i \\ &- \sum_{\{i:y_i>0\}} \frac{\varphi(\mathbf{z}_i'\boldsymbol{\gamma})}{\left[1 - \Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]} \mathbf{z}_i \end{aligned}$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} &= \sum_{\{i:y_i=0\}} \frac{-\left[1 - \Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\exp(\mathbf{x}_i'\boldsymbol{\beta})\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))}{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1 - \Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))} \mathbf{x}_i \\ &+ \sum_{\{i:y_i>0\}} \left[y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta}) \right] \mathbf{x}_i \end{aligned}$$

Zero-Inflated Negative Binomial Regression

The zero-inflated negative binomial (ZINB) model in PROC HPCOUNTREG is based on the negative binomial model that has a quadratic variance function (when DIST=NEGBIN in the MODEL or PROC HPCOUNTREG statement). The ZINB model is obtained by specifying a negative binomial distribution for the data generation process referred to earlier as Process 2:

$$g(y_i) = \frac{\Gamma(y_i + \alpha^{-1})}{y_i!\Gamma(\alpha^{-1})} \left(\frac{\alpha^{-1}}{\alpha^{-1} + \mu_i} \right)^{\alpha^{-1}} \left(\frac{\mu_i}{\alpha^{-1} + \mu_i} \right)^{y_i}$$

Thus the ZINB model is defined to be

$$\begin{aligned} P(y_i = 0|\mathbf{x}_i, \mathbf{z}_i) &= F_i + (1 - F_i)\,(1 + \alpha\mu_i)^{-\alpha^{-1}} \\ P(y_i|\mathbf{x}_i, \mathbf{z}_i) &= (1 - F_i)\,\frac{\Gamma(y_i + \alpha^{-1})}{y_i!\Gamma(\alpha^{-1})}\left(\frac{\alpha^{-1}}{\alpha^{-1} + \mu_i}\right)^{\alpha^{-1}} \\ &\times \left(\frac{\mu_i}{\alpha^{-1} + \mu_i}\right)^{y_i}, \quad y_i > 0 \end{aligned}$$

In this case, the conditional expectation (E) and conditional variance (V) of y_i are

$$E(y_i|\mathbf{x}_i, \mathbf{z}_i) = \mu_i(1 - F_i)$$

$$V(y_i|\mathbf{x}_i, \mathbf{z}_i) = E(y_i|\mathbf{x}_i, \mathbf{z}_i)\,[1 + \mu_i(F_i + \alpha)]$$

Like the ZIP model, the ZINB model exhibits overdispersion because the conditional variance exceeds the conditional mean.

ZINB Model with Logistic Link Function

In this model, the probability φ_i is given by the logistic function, namely

$$\varphi_i = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

The log-likelihood function is

$$\begin{aligned} \mathcal{L} &= \sum_{\{i:y_i=0\}} \ln\left[\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + (1 + \alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}\right] \\ &+ \sum_{\{i:y_i>0\}} \sum_{j=0}^{y_i-1} \ln(j + \alpha^{-1}) \\ &+ \sum_{\{i:y_i>0\}} \{-\ln(y_i!) - (y_i + \alpha^{-1})\ln(1 + \alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) + y_i\ln(\alpha) + y_i\mathbf{x}_i'\boldsymbol{\beta}\} \\ &- \sum_{i=1}^{N} \ln\left[1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})\right] \end{aligned}$$

The gradient for this model is given by

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\boldsymbol{\gamma}} &= \sum_{\{i:y_i=0\}} \left[\frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + (1 + \alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}}\right]\mathbf{z}_i \\ &- \sum_{i=1}^{N}\left[\frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma})}\right]\mathbf{z}_i \end{aligned}$$

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} &= \sum_{\{i:y_i=0\}} \left[\frac{-\exp(\mathbf{x}_i'\boldsymbol{\beta})(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}-1}}{\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + (1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}} \right] \mathbf{x}_i \\
&+ \sum_{\{i:y_i>0\}} \left[\frac{y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta})}{1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})} \right] \mathbf{x}_i
\end{aligned}
$$

$$
\frac{\partial \mathcal{L}}{\partial \alpha} = \sum_{\{i:y_i=0\}} \frac{\alpha^{-2}\left[(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) - \alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})\right]}{\exp(\mathbf{z}_i'\boldsymbol{\gamma})(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{(1+\alpha)/\alpha} + (1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))}
$$

$$
+ \sum_{\{i:y_i>0\}} \left\{ -\alpha^{-2} \sum_{j=0}^{y_i-1} \frac{1}{(j+\alpha^{-1})} + \alpha^{-2}\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) + \frac{y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta})}{\alpha(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))} \right\}
$$

ZINB Model with Standard Normal Link Function

For this model, the probability φ_i is expressed by the standard normal distribution function (probit function): $\varphi_i = \Phi(\mathbf{z}_i'\boldsymbol{\gamma})$. The log-likelihood function is

$$
\begin{aligned}
\mathcal{L} &= \sum_{\{i:y_i=0\}} \ln\left\{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right](1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}\right\} \\
&+ \sum_{\{i:y_i>0\}} \ln\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right] \\
&+ \sum_{\{i:y_i>0\}} \sum_{j=0}^{y_i-1} \{\ln(j+\alpha^{-1})\} \\
&- \sum_{\{i:y_i>0\}} \ln(y_i!) \\
&- \sum_{\{i:y_i>0\}} (y_i+\alpha^{-1})\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) \\
&+ \sum_{\{i:y_i>0\}} y_i\ln(\alpha) \\
&+ \sum_{\{i:y_i>0\}} y_i\mathbf{x}_i'\boldsymbol{\beta}
\end{aligned}
$$

The gradient for this model is given by

$$
\frac{\partial \mathcal{L}}{\partial \boldsymbol{\gamma}} = \sum_{\{i:y_i=0\}} \left[\frac{\varphi(\mathbf{z}_i'\boldsymbol{\gamma})\left[1-(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}\right]}{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right](1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}} \right] \mathbf{z}_i
$$

$$- \sum_{\{i:y_i>0\}} \left[\frac{\varphi(\mathbf{z}_i'\boldsymbol{\gamma})}{1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})} \right] \mathbf{z}_i$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = \sum_{\{i:y_i=0\}} \frac{-\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right] \exp(\mathbf{x}_i'\boldsymbol{\beta})(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-(1+\alpha)/\alpha}}{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right](1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}} \mathbf{x}_i$$

$$+ \sum_{\{i:y_i>0\}} \left[\frac{y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta})}{1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})} \right] \mathbf{x}_i$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \alpha} &= \sum_{\{i:y_i=0\}} \frac{\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\alpha^{-2}\left[(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) - \alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})\right]}{\Phi(\mathbf{z}_i'\boldsymbol{\gamma})(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{(1+\alpha)/\alpha} + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right](1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))} \\ &+ \sum_{\{i:y_i>0\}} \left\{ -\alpha^{-2} \sum_{j=0}^{y_i-1} \frac{1}{(j+\alpha^{-1})} + \alpha^{-2}\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) + \frac{y_i - \exp(\mathbf{x}_i'\boldsymbol{\beta})}{\alpha(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))} \right\} \end{aligned}$$

Computational Resources

The time and memory that PROC HPCOUNTREG requires are proportional to the number of parameters in the model and the number of observations in the data set being analyzed. Less time and memory are required for smaller models and fewer observations. When PROC HPCOUNTREG is run in the high-performance distributed environment, the amount of time required is also affected by the number of nodes and the number of threads per node as specified in the PERFORMANCE statement.

The method that is chosen to calculate the variance-covariance matrix and the optimization method also affect the time and memory resources. All optimization methods available through the METHOD= option have similar memory use requirements. The processing time might differ for each method, depending on the number of iterations and functional calls needed. The data set is read into memory to save processing time. If not enough memory is available to hold the data, the HPCOUNTREG procedure stores the data in a utility file on disk and rereads the data as needed from this file, substantially increasing the execution time of the procedure. The gradient and the variance-covariance matrix must be held in memory. If the model has p parameters including the intercept, then at least $8 * (p + p * (p + 1)/2)$ bytes of memory are needed. The processing time is also a function of the number of iterations needed to converge to a solution for the model parameters. The number of iterations that are needed cannot be known in advance. You can use the MAXITER= option to limit the number of iterations that PROC HPCOUNTREG executes. You can alter the convergence criteria by using the nonlinear optimization options available in the PROC HPCOUNTREG statement. For a list of all the nonlinear optimization options, see “Optimization Control Options” on page 46.

Covariance Matrix Types

The COVEST= option in the PROC HPCOUNTREG statement enables you to specify the estimation method for the covariance matrix. COVEST=HESSIAN estimates the covariance matrix that is based on the inverse of the Hessian matrix; COVEST=OP uses the outer product of gradients; and COVEST=QML produces the covariance matrix that is based on both the Hessian and outer product matrices. Although all three methods produce asymptotically equivalent results, they differ in computational intensity and produce results that might differ in finite samples. The COVEST=OP option provides the covariance matrix that is typically the easiest to compute. In some cases, the OP approximation is considered more efficient than the Hessian or QML approximation because it contains fewer random elements. The QML approximation is computationally the most complex because it requires both the outer product of gradients and the Hessian matrix. In most cases, the OP or Hessian approximation is preferred to QML. The need for QML approximation arises in cases where the model is misspecified and the information matrix equality does not hold. The default is COVEST=HESSIAN.

Displayed Output

PROC HPCOUNTREG produces the following displayed output.

Model Fit Summary

The “Model Fit Summary” table contains the following information:

- dependent (count) variable name
- number of observations used
- number of missing values in data set, if any
- data set name
- type of model that was fit
- offset variable name, if any
- zero-inflated link function, if any
- zero-inflated offset variable name, if any
- log-likelihood value at solution
- maximum absolute gradient at solution
- number of iterations
- AIC value at solution (smaller value indicates better fit)
- SBC value at solution (smaller value indicates better fit)

A line in the “Model Fit Summary” table indicates whether the algorithm successfully converged.

Parameter Estimates

The "Parameter Estimates" table in the displayed output gives the estimates for the ZI intercept and ZI explanatory variables; they are labeled with the prefix "Inf_". For example, the ZI intercept is labeled "Inf_intercept". If you specify "Age" (a variable in your data set) as a ZI explanatory variable, then the "Parameter Estimates" table labels the corresponding parameter estimate "Inf_Age". If you do not list any ZI explanatory variables (for the ZI option VAR=), then only the intercept term is estimated.

"_Alpha" is the negative binomial dispersion parameter. The t statistic that is given for "_Alpha" is a test of overdispersion.

Covariance of Parameter Estimates

If you specify the COVB option in the PROC HPCOUNTREG or MODEL statement, the HPCOUNTREG procedure displays the estimated covariance matrix, which is defined as the inverse of the information matrix at the final iteration.

Correlation of Parameter Estimates

If you specify the CORRB option in the PROC HPCOUNTREG or MODEL statement, the HPCOUNTREG procedure displays the estimated correlation matrix, which is based on the Hessian matrix used at the final iteration.

OUTPUT OUT= Data Set

The OUTPUT statement creates a new SAS data set that contains various estimates that you specify. You can request that the output data set contain the estimates of $\mathbf{x}_i'\boldsymbol{\beta}$, the expected value of the response variable, and the probability that the response variable will take the current value. Furthermore, if a zero-inflated model is fit, you can request that the output data set contain the estimates of $\mathbf{z}_i'\boldsymbol{\gamma}$ and the probability that the response is 0 as a result of the zero-generating process. These statistics can be computed for all observations in which the regressors are not missing, even if the response is missing. By adding observations with missing response values to the input data set, you can compute these statistics for new observations or for settings of the regressors that are not present in the data without affecting the model fit. Because of potential space limitations on the client workstation, the data set that is created by the OUTPUT statement does not contain the variables in the input data set.

OUTEST= Data Set

The OUTEST= data set is made up of at least two rows: the first row (with _TYPE_='PARM') contains each of the parameter estimates in the model, and the second row (with _TYPE_='STD') contains the standard errors for the parameter estimates in the model.

If you use the COVOUT option in the PROC HPCOUNTREG statement, the OUTEST= data set also contains the covariance matrix for the parameter estimates. The covariance matrix appears in the observations with _TYPE_='COV', and the _NAME_ variable labels the rows with the parameter names.

ODS Table Names

PROC HPCOUNTREG assigns a name to each table that it creates. You can use these names to denote the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These table names are listed in Table 3.2.

Table 3.2 ODS Tables Produced in PROC HPCOUNTREG

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
FitSummary	Summary of nonlinear estimation	Default
ConvergenceStatus	Convergence status	Default
ParameterEstimates	Parameter estimates	Default
CovB	Covariance of parameter estimates	COVB
CorrB	Correlation of parameter estimates	CORRB

Examples: The HPCOUNTREG Procedure

Example 3.1: High-Performance Zero-Inflated Poisson Model

This example shows the use of the HPCOUNTREG procedure with an emphasis on large data set processing and the performance improvements that are achieved by executing in the high-performance distributed environment.

The following DATA step generates one million replicates from the zero-inflated Poisson (ZIP) model. The model contains seven variables and three variables that correspond to the zero-inflated process.

```
data simulate;
   call streaminit(12345);
   array vars x1-x7;
   array zero_vars z1-z3;

   array parms{7}  (.3 .4 .2 .4 -.3 -.5 -.3);
   array zero_parms{3} (-.6 .3 .2);

   intercept=2;
   z_intercept=-1;
   theta=0.5;

   do i=1 to 1000000;
      sum_xb=0;
      sum_gz=0;
```

```
         do j=1 to 7;
            vars[j]=rand('NORMAL',0,1);
            sum_xb=sum_xb+parms[j]*vars[j];
         end;
         mu=exp(intercept+sum_xb);
         y_p=rand('POISSON', mu);

         do j=1 to 3;
            zero_vars[j]=rand('NORMAL',0,1);
            sum_gz = sum_gz+zero_parms[j]*zero_vars[j];
         end;
         z_gamma = z_intercept+sum_gz;
         pzero = cdf('LOGISTIC',z_gamma);
         cut=rand('UNIFORM');
         if cut<pzero then y_p=0;
         output;
      end;
   keep y_p x1-x7 z1-z3;
   run;
```

The following statements estimate a zero-inflated Poisson model. The model is executed in the distributed computing environment on two threads and only one node. These settings are used to obtain a hypothetical environment that might resemble running the HPCOUNTREG procedure on a desktop workstation with a dual-core CPU. To run these statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to the macro variables in the example with the appropriate values.

```
   option set=GRIDHOST="&GRIDHOST";
   option set=GRIDINSTALLLOC="&GRIDINSTALLLOC";

proc hpcountreg data=simulate dist=zip;
   performance nthreads=2 nodes=1 details;
   model y_p=x1-x7;
   zeromodel y_p ~ z1-z3;
run;
```

Output 3.1.1 shows the results for the zero-inflated Poisson model. The “Performance Information” table shows that the model was estimated on the grid defined in a macro variable named GRIDHOST in a distributed environment on only one node, and it shows that two threads were employed. The “Model Fit Summary” table shows detailed information about the model and indicates that all one million observations were used to fit the model. All parameter estimates in the “Parameter Estimates” table are highly significant and correspond to their theoretical values set during the data generating process. The optimization of the model that contains one million observations took 46.47 seconds.

Output 3.1.1 Zero-Inflated Poisson Model Execution on One Node and Two Threads

```
                         The HPCOUNTREG Procedure

                         Performance Information

              Host Node                      << your grid host >>
              Execution Mode                 Distributed
              Grid Mode                      Symmetric
              Number of Compute Nodes        1
              Number of Threads per Node     2


                            Model Fit Summary

                  Dependent Variable                         y_p
                  Number of Observations                 1000000
                  Data Set                         WORK.SIMULATE
                  Model                                      ZIP
                  ZI Link Function                      Logistic
                  Log Likelihood                        -2215238
                  Maximum Absolute Gradient           2.0586E-8
                  Number of Iterations                         7
                  Optimization Method             Newton-Raphson
                  AIC                                    4430500
                  SBC                                    4430642


             Convergence criterion (FCONV=2.220446E-16) satisfied.


                           Parameter Estimates

                                                 Standard
     Parameter            DF     Estimate           Error     t Value    Pr > |t|

     Intercept             1       2.0005        0.000492     4069.80      <.0001
     x1                    1       0.2995        0.000352      850.17      <.0001
     x2                    1       0.3998        0.000353     1132.23      <.0001
     x3                    1       0.2008        0.000352      570.27      <.0001
     x4                    1       0.3994        0.000353     1132.85      <.0001
     x5                    1      -0.2995        0.000353     -848.95      <.0001
     x6                    1      -0.5000        0.000353     -1414.9      <.0001
     x7                    1      -0.3002        0.000352     -852.14      <.0001
     Inf_Intercept         1      -0.9993        0.002521     -396.45      <.0001
     Inf_z1                1      -0.6024        0.002585     -233.02      <.0001
     Inf_z2                1       0.2976        0.002454      121.25      <.0001
     Inf_z3                1       0.1974        0.002430       81.20      <.0001
```

Output 3.1.1 *continued*

```
                          Procedure Task Timing

        Task                                         Seconds     Percent

        Reading and Levelizing Data                     0.43       0.91%
        Communication to Client                         0.00       0.01%
        Optimization                                   46.47      98.91%
        Post-Optimization                               0.08       0.17%
```

In the following statements, the PERFORMANCE statement is modified to use a grid with 10 nodes, with each node capable of spawning eight threads:

```
proc hpcountreg data=simulate dist=zip;
   performance nthreads=8 nodes=10 details;
   model y_p=x1-x7;
   zeromodel y_p ~ z1-z3;
run;
```

Because the two models being estimated are identical, it is reasonable to expect that Output 3.1.1 and Output 3.1.2 would show the same results. However, you can see a significant difference in performance between the two models. The second model, which was run on a grid that used 10 nodes with eight threads each, took only 2.49 seconds instead of 46.47 seconds to optimize.

In certain circumstances, you might observe slight numerical differences in the results, depending on the number of nodes and threads involved. This happens because the order in which partial results are accumulated can make a difference in the final result, owing to the limits of numerical precision and the propagation of error in numerical computations.

Output 3.1.2 Zero-Inflated Poisson Model Execution on 10 Nodes with Eight Threads Each

```
                         The HPCOUNTREG Procedure

                         Performance Information

               Host Node                     << your grid host >>
               Execution Mode                Distributed
               Grid Mode                     Symmetric
               Number of Compute Nodes       10
               Number of Threads per Node    8
```

Output 3.1.2 *continued*

```
                         Model Fit Summary

             Dependent Variable                         y_p
             Number of Observations                 1000000
             Data Set                         WORK.SIMULATE
             Model                                      ZIP
             ZI Link Function                      Logistic
             Log Likelihood                        -2215238
             Maximum Absolute Gradient            2.0608E-8
             Number of Iterations                         7
             Optimization Method             Newton-Raphson
             AIC                                    4430500
             SBC                                    4430642


        Convergence criterion (FCONV=2.220446E-16) satisfied.


                        Parameter Estimates

                                         Standard
Parameter           DF    Estimate          Error    t Value    Pr > |t|

Intercept            1      2.0005       0.000492    4069.80      <.0001
x1                   1      0.2995       0.000352     850.17      <.0001
x2                   1      0.3998       0.000353    1132.23      <.0001
x3                   1      0.2008       0.000352     570.27      <.0001
x4                   1      0.3994       0.000353    1132.85      <.0001
x5                   1     -0.2995       0.000353    -848.95      <.0001
x6                   1     -0.5000       0.000353    -1414.9      <.0001
x7                   1     -0.3002       0.000352    -852.14      <.0001
Inf_Intercept        1     -0.9993       0.002521    -396.45      <.0001
Inf_z1               1     -0.6024       0.002585    -233.02      <.0001
Inf_z2               1      0.2976       0.002454     121.25      <.0001
Inf_z3               1      0.1974       0.002430      81.20      <.0001


                       Procedure Task Timing

  Task                                          Seconds     Percent

  Reading and Levelizing Data                      0.04       1.58%
  Communication to Client                          0.07       2.67%
  Optimization                                     2.49      90.09%
  Post-Optimization                                0.16       5.66%
```

As this example suggests, increasing the number of nodes and the number of threads per node improves performance significantly. When you use the parallelism afforded by a high-performance distributed environment, you can see an even more dramatic reduction in the time required for the optimization as the number of observations in the data set increases. When the data set is extremely large, the computations might not even be possible in some cases, given the typical memory resources and computational constraints of a desktop computer. Under such circumstances the high-performance distributed environment becomes a necessity.

References

Cameron, A. C. and Trivedi, P. K. (1986), “Econometric Models Based on Count Data: Comparisons and Applications of Some Estimators and Some Tests,” *Journal of Applied Econometrics*, 1, 29–53.

Cameron, A. C. and Trivedi, P. K. (1998), *Regression Analysis of Count Data*, Cambridge: Cambridge University Press.

LaMotte, L. R. (1994), “A Note on the Role of Independence in t Statistics Constructed from Linear Statistics in Regression Models,” *The American Statistician*, 48, 238–240.

Long, J. S. (1997), *Regression Models for Categorical and Limited Dependent Variables*, Thousand Oaks, CA: Sage Publications.

Chapter 4
The HPQLIM Procedure

Contents

Overview: HPQLIM Procedure

The HPQLIM (high-performance qualitative and limited dependent variable model) procedure is a high-performance version of the QLIM procedure in SAS/ETS software, which analyzes univariate limited dependent variable models in which dependent variables are observed only in a limited range of values. Unlike the QLIM procedure, which can be run only on an individual workstation, the HPQLIM procedure takes advantage of a computing environment that enables it to distribute the optimization task to one or more nodes. In addition, each node can use one or more threads to perform the optimization on its subset of the data. When several nodes are used and each node uses several threads to carry out its part of the work, the result is a highly parallel computation that provides a dramatic gain in performance.

With the HPQLIM procedure you can read and write data in distributed form and perform analyses in distributed mode and single-machine mode. For more information about how to affect the execution mode of SAS high-performance analytical procedures, see the section "Processing Modes" on page 6 in Chapter 2, "Shared Concepts and Topics."

The HPQLIM procedure is specifically designed to operate in the high-performance distributed environment. It can use maximum likelihood or Bayesian methods. In both cases, the likelihood evaluation is performed in a distributed environment. By default, PROC HPQLIM uses multiple threads to perform computations.

The HPQLIM procedure is similar in use to the other SAS procedures that support regression or simultaneous equations models. For example, the standard model with censoring or truncation is estimated by specifying the endogenous variable to be truncated or censored. When the data are limited by specific values or variables, the limits of the dependent variable can be specified with the CENSORED or TRUNCATED option in the ENDOGENOUS or MODEL statement. For example, the two-limit censored model requires two variables: one that contains the lower (bottom) bound and one that contains the upper (top) bound. The following statements execute the model in the distributed computing environment with two threads and four nodes:

```
proc hpqlim data=a;
   model y = x1 x2 x3;
   endogenous y ~ censored(lb=bottom ub=top);
   performance nthreads=2 nodes=4 details;
run;
```

The bounds can be numbers if they are fixed for all observations in the data set. For example, the standard Tobit model can be specified as follows:

```
proc hpqlim data=a;
   model y = x1 x2 x3;
   endogenous y ~ censored(lb=0);
   performance nthreads=2 nodes=4 details;
run;
```

PROC HPQLIM Features

The HPQLIM procedure supports the following models:

- linear regression models with heteroscedasticity
- Tobit models (censored and truncated) with heteroscedasticity
- stochastic frontier production and cost models

In linear regression models with heteroscedasticity, the assumption that error variance is constant across observations is relaxed. The HPQLIM procedure allows for a number of different linear and nonlinear variance specifications.

The HPQLIM procedure also offers a class of models in which the dependent variable is censored or truncated from below or above or both. When a continuous dependent variable is observed only within a certain range, and values outside this range are not available, the HPQLIM procedure offers a class of models that adjust for truncation. In some cases, the dependent variable is continuous only in a certain range, and all values outside this range are reported as being on its boundary. For example, if it is not possible to observe negative values, the value of the dependent variable is reported as equal to 0. Because the data are censored, ordinary least squares (OLS) results are inconsistent, and it cannot be guaranteed that the predicted values from the model will fall in the appropriate region.

Stochastic frontier production and cost models allow for random shocks of the production or cost. They include a systematic positive component in the error term that adjusts for technical or cost inefficiency.

The HPQLIM procedure can use maximum likelihood or Bayesian methods. Initial starting values for the nonlinear optimizations are typically calculated by OLS. Initial values for the Bayesian sampling are typically calculated by maximum likelihood.

Getting Started: HPQLIM Procedure

This example illustrates the use of the HPQLIM procedure. The data were originally published by Mroz (1987), and the following statements show a subset of the Mroz (1987) data set:

```
title1 'Estimating a Tobit model';

data subset;
   input Hours Yrs_Ed Yrs_Exp @@;
   if Hours eq 0 then Lower=.;
      else               Lower=Hours;
datalines;
0 8 9 0 8 12 0 9 10 0 10 15 0 11 4 0 11 6
1000 12 1 1960 12 29 0 13 3 2100 13 36
3686 14 11 1920 14 38 0 15 14 1728 16 3
1568 16 19 1316 17 7 0 17 15
;
```

In these data, Hours is the number of hours that the wife worked outside the household in a given year, Yrs_Ed is the years of education, and Yrs_Exp is the years of work experience.

By the nature of the data it is clear that there are a number of women who committed some positive number of hours to outside work ($y_i > 0$ is observed). There are also a number of women who did not work outside the home at all ($y_i = 0$ is observed). This yields the following model:

$$y_i^* = \mathrm{x}_i' \boldsymbol{\beta} + \epsilon_i$$

$$y_i = \begin{cases} y_i^* & \text{if } y_i^* > 0 \\ 0 & \text{if } y_i^* \leq 0 \end{cases}$$

where $\epsilon_i \sim iidN(0, \sigma^2)$ and the set of explanatory variables is denoted by $\mathbf{x}_i$. The following statements fit a Tobit model to the hours worked with years of education and years of work experience as covariates:

```
/*-- Tobit Model --*/
proc hpqlim data=subset;
   model hours = yrs_ed yrs_exp;
   endogenous hours ~ censored(lb=0);
   performance nthreads=2 nodes=4 details;
run;
```

The output of the HPQLIM procedure is shown in Output 4.1.

Figure 4.1 Tobit Analysis Results

```
                          Estimating a Tobit model

                            The HPQLIM Procedure

                             Model Fit Summary

            Number of Endogenous Variables                1
            Endogenous Variable                       Hours
            Number of Observations                       17
            Log Likelihood                        -74.93700
            Maximum Absolute Gradient            1.18953E-6
            Number of Iterations                         23
            Optimization Method                Quasi-Newton
            AIC                                   157.87400
            Schwarz Criterion                     161.20685


                            Parameter Estimates

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

      Intercept     1    -5598.295130    27.692220    -202.16      <.0001
      Yrs_Ed        1      373.123254    53.988877       6.91      <.0001
      Yrs_Exp       1       63.336247    36.551299       1.73      0.0831
      _Sigma        1     1582.859635   390.076480       4.06      <.0001
```

The “Parameter Estimates” table contains four rows. The first three rows correspond to the vector estimate of the regression coefficients $\boldsymbol{\beta}$. The last row is called _Sigma, which corresponds to the estimate of the error variance σ.

Syntax: HPQLIM Procedure

The following statements are available in the HPQLIM procedure:

> **PROC HPQLIM** *options* ;
> **BAYES** <*options*> ;
> **BOUNDS** *bound1* < , *bound2* . . . > ;
> **FREQ** *variable* ;
> **ENDOGENOUS** *variables* ∼ *options* ;
> **HETERO** *dependent variables* ∼ *exogenous variables / options* ;
> **INIT** *initvalue1* < , *initvalue2* . . . > ;
> **MODEL** *dependent variables* = *regressors / options* ;
> **OUTPUT** *options* ;
> **PRIOR** *variables* ∼ *distributions* ;
> **RESTRICT** *restriction1* < , *restriction2* . . . > ;
> **TEST** *options* ;
> **WEIGHT** *variable* ;

One MODEL statement is required. If a FREQ or WEIGHT statement is specified more than once, the variable that is specified in the first instance is used.

Functional Summary

Table 4.1 summarizes the statements and options used with the HPQLIM procedure.

Table 4.1 HPQLIM Functional Summary

Description	Statement	Option
Data Set Options		
Specifies the input data set	HPQLIM	DATA=
Writes parameter estimates to an output data set	HPQLIM	OUTEST=
Writes predictions to an output data set	OUTPUT	OUT=
Declaring the Role of Variables		
Specifies a frequency variable	FREQ	
Specifies a weight variable	WEIGHT	NONORMALIZE
Printing Control Options		
Requests all printing options	HPQLIM	PRINTALL
Prints the correlation matrix of the estimates	HPQLIM	CORRB
Prints the covariance matrix of the estimates	HPQLIM	COVB
Suppresses the normal printed output	HPQLIM	NOPRINT
Plotting Options		
Displays plots	HPQLIM	PLOTS=

Table 4.1 *continued*

Description	Statement	Option
Optimization Process Control Options		
Selects the iterative minimization method to use	HPQLIM	METHOD=
Specifies the maximum number of iterations allowed	HPQLIM	MAXITER=
Specifies the maximum number of function calls	HPQLIM	MAXFUNC=
Specifies the upper limit of CPU time in seconds	HPQLIM	MAXTIME=
Specifies an absolute convergence criterion	HPQLIM	ABSCONV=
Specifies an absolute function convergence criterion	HPQLIM	ABSFCONV=
Specifies an absolute gradient convergence criterion	HPQLIM	ABSGCONV=
Specifies a relative function convergence criterion	HPQLIM	FCONV=
Specifies a relative gradient convergence criterion	HPQLIM	GCONV=
Specifies an absolute parameter convergence criterion	HPQLIM	ABSXCONV=
Specifies a matrix singularity criterion	HPQLIM	SINGULAR=
Sets boundary restrictions on parameters	BOUNDS	
Sets initial values for parameters	INIT	
Sets linear restrictions on parameters	RESTRICT	
Model Estimation Options		
Suppresses the intercept parameter	MODEL	NOINT
Specifies the method to calculate parameter covariance	HPQLIM	COVEST=
Bayesian MCMC Options		
Specifies the initial values of the MCMC	INIT	
Specifies the maximum number of tuning phases	BAYES	MAXTUNE=
Specifies the minimum number of tuning phases	BAYES	MINTUNE=
Specifies the number of burn-in iterations	BAYES	NBI=
Specifies the number of iterations during the sampling phase	BAYES	NMC=
Specifies the number of iterations during the tuning phase	BAYES	NTU=
Controls options for constructing the initial proposal covariance matrix	BAYES	PROPCOV
Specifies the sampling scheme	BAYES	SAMPLING=
Specifies the random number generator seed	BAYES	SEED=
Controls the thinning of the Markov chain	BAYES	THIN=
Bayesian Summary Statistics and Convergence Diagnostic Options		
Displays convergence diagnostics	BAYES	DIAGNOSTICS=
Displays summary statistics of the posterior samples	BAYES	STATISTICS=
Bayesian Prior and Posterior Sample Options		
Specifies a SAS data set for the posterior samples	BAYES	OUTPOST=

Table 4.1 *continued*

Description	Statement	Option
Bayesian Analysis Options		
Specifies the normal prior distribution	PRIOR	NORMAL(MEAN=, VAR=)
Specifies the gamma prior distribution	PRIOR	GAMMA(SHAPE=, SCALE=)
Specifies the inverse gamma prior distribution	PRIOR	IGAMMA(SHAPE=, SCALE=)
Specifies the uniform prior distribution	PRIOR	UNIFORM(MIN=, MAX=)
Specifies the beta prior distribution	PRIOR	BETA(SHAPE1=, SHAPE2=, MIN=, MAX=)
Specifies the *t* prior distribution	PRIOR	T(LOCATION=, DF=)
Endogenous Variable Options		
Specifies a censored variable	ENDOGENOUS	CENSORED()
Specifies a truncated variable	ENDOGENOUS	TRUNCATED()
Specifies a stochastic frontier variable	ENDOGENOUS	FRONTIER()
Heteroscedasticity Model Options		
Specifies the function for heteroscedasticity models	HETERO	LINK=
Squares the function for heteroscedasticity models	HETERO	SQUARE
Specifies no constant for heteroscedasticity models	HETERO	NOCONST
Output Control Options		
Outputs predicted values	OUTPUT	PREDICTED
Outputs the structured part	OUTPUT	XBETA
Outputs residuals	OUTPUT	RESIDUAL
Outputs the error standard deviation	OUTPUT	ERRSTD
Outputs marginal effects	OUTPUT	MARGINAL
Outputs the expected value	OUTPUT	EXPECTED
Outputs the conditional expected value	OUTPUT	CONDITIONAL
Outputs technical efficiency measures	OUTPUT	TE1
	OUTPUT	TE2
Includes covariances in the OUTEST= data set	HPQLIM	COVOUT
Includes correlations in the OUTEST= data set	HPQLIM	CORROUT
Test Request Options		
Requests Wald, Lagrange multiplier, and likelihood ratio tests	TEST	ALL
Requests the Wald test	TEST	WALD
Requests the Lagrange multiplier test	TEST	LM
Requests the likelihood ratio test	TEST	LR

PROC HPQLIM Statement

PROC HPQLIM *options* **;**

The PROC HPQLIM statement invokes the HPQLIM procedure. You can specify the following *options*.

Data Set Options

DATA=*SAS-data-set*
specifies the input SAS data set. If this option is not specified, PROC HPQLIM uses the most recently created SAS data set.

Output Data Set Options

OUTEST=*SAS-data-set*
writes the parameter estimates to an output data set.

COVOUT
writes the covariance matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

CORROUT
writes the correlation matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

Printing Options

NOPRINT
suppresses the normal printed output but does not suppress error listings. If this option is specified, then any other print option is turned off.

PRINTALL
turns on all the printing options. The options that are set by PRINTALL are COVB and CORRB.

CORRB
prints the correlation matrix of the parameter estimates.

COVB
prints the covariance matrix of the parameter estimates.

Model Estimation Options

COVEST=*covariance-option*
specifies the method for calculating the covariance matrix of parameter estimates. You can specify the following *covariance-options*.

OP	specifies the covariance from the outer product matrix.
HESSIAN	specifies the covariance from the inverse Hessian matrix.

QML specifies the covariance from the outer product and Hessian matrices (the quasi-maximum likelihood estimates).

The default is COVEST=HESSIAN.

Optimization Control Options

PROC HPQLIM uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks. You can specify the following *options*:

ABSCONV=*r*

ABSTOL=*r*

specifies an absolute function value convergence criterion by which minimization stops when $f(\theta^{(k)}) \leq r$. The default value of *r* is the negative square root of the largest double-precision value, which serves only as a protection against overflows.

ABSFCONV=*r*

ABSFTOL=*r*

specifies an absolute function difference convergence criterion by which minimization stops when the function value has a small change in successive iterations:

$$|f(\theta^{(k-1)}) - f(\theta^{(k)})| \leq r$$

The default value is $r = 0$.

ABSGCONV=*r*

ABSGTOL=*r*

specifies an absolute gradient convergence criterion. Optimization stops when the maximum absolute gradient element is small:

$$\max_j |g_j(\theta^{(k)})| \leq r$$

The default value is *r*=1E–5.

ABSXCONV=*r*

ABSXTOL=*r*

specifies an absolute parameter convergence criterion. Optimization stops when the Euclidean distance between successive parameter vectors is small:

$$\parallel \theta^{(k)} - \theta^{(k-1)} \parallel_2 \leq r$$

The default is 0.

FCONV=*r*

FTOL=*r*

specifies a relative function convergence criterion. Optimization stops when a relative change of the function value in successive iterations is small:

$$\frac{|f(\theta^{(k)}) - f(\theta^{(k-1)})|}{|f(\theta^{(k-1)})|} \leq r$$

The default value is $r = 2\epsilon$, where ϵ denotes the machine precision constant, which is the smallest double-precision floating-point number such that $1 + \epsilon > 1$.

GCONV=*r*

GTOL=*r*

specifies a relative gradient convergence criterion. For all techniques except CONGRA, optimization stops when the normalized predicted function reduction is small:

$$\frac{g(\theta^{(k)})^T [H^{(k)}]^{-1} g(\theta^{(k)})}{|f(\theta^{(k)})|} \leq r$$

For the CONGRA technique (where a reliable Hessian estimate H is not available), the following criterion is used:

$$\frac{\| g(\theta^{(k)}) \|_2^2 \quad \| s(\theta^{(k)}) \|_2}{\| g(\theta^{(k)}) - g(\theta^{(k-1)}) \|_2 \; |f(\theta^{(k)})|} \leq r$$

The default value is r =1E–8.

MAXFUNC=*i*

MAXFU=*i*

specifies the maximum number of function calls in the optimization process. The default is 1,000.

The optimization can terminate only after completing a full iteration. Therefore, the number of function calls that are actually performed can exceed the number of calls that are specified by this option.

MAXITER=*i*

MAXIT=*i*

specifies the maximum number of iterations in the optimization process. The default is 200.

MAXTIME=*r*

specifies an upper limit of *r* seconds of CPU time for the optimization process. The default value is the largest floating-point double representation of your computer. The time that is specified by this option is checked only once at the end of each iteration. Therefore, the actual running time can be much longer than *r*. The actual running time includes the remaining time needed to finish the iteration and the time needed to generate the output of the results.

METHOD=*value*

specifies the iterative minimization method to use. The default is METHOD=NEWRAP. You can specify the following *values*:

CONGRA	specifies the conjugate-gradient method.
DBLDOG	specifies the double dogleg method.
NONE	specifies that no optimization be performed beyond using the ordinary least squares method to compute the parameter estimates.
NEWRAP	specifies the Newton-Raphson method (the default).
NRRIDG	specifies the Newton-Raphson Ridge method.
QUANEW	specifies the quasi-Newton method.
TRUREG	specifies the trust region method.

SINGULAR=*r*

specifies the general singularity criterion that is applied by the HPQLIM procedure in sweeps and inversions. The default for the optimization is 1E–8.

Plotting Options

PLOTS< (*global-plot-options*) > = *plot-request* | *(plot-requests)*

controls the display of plots. By default, the plots are displayed in panels unless the UNPACK *global-plot-option* is specified. When you specify only one *plot-request*, you can omit the parentheses around it.

Global Plot Options

You can specify the following *global-plot-options*:

ONLY

displays only the requested plot.

UNPACKPANEL

UNPACK

specifies that all paneled plots be unpacked, meaning that each plot in a panel is displayed separately.

Plot Requests

You can specify the following *plot-requests*:

ALL

specifies all types of available plots.

AUTOCORR< (LAGS=*n*) >

displays the autocorrelation function plots for the parameters. The optional LAGS= suboption specifies the number (up to lag *n*) of autocorrelations to be plotted in the autocorrelation function plot. If this suboption is not specified, autocorrelations are plotted up to lag 50. This *plot-request* is available only for Bayesian analysis.

BAYESDIAG

is equivalent to specifying the TRACE, AUTOCORR, and DENSITY *plot-requests.*

DENSITY< (FRINGE) >

displays the kernel density plots for the parameters. If you specify the FRINGE suboption, a fringe plot is created on the X axis of the kernel density plot. This *plot-request* is available only for Bayesian analysis.

NONE

suppresses all diagnostic plots.

TRACE< (SMOOTH) >

displays the trace plots for the parameters. The SMOOTH suboption displays a fitted penalized B-spline curve for each plot. This *plot-request* is available only for Bayesian analysis.

BAYES Statement

BAYES < *options* > **;**

The BAYES statement controls the Metropolis sampling scheme that is used to obtain samples from the posterior distribution of the underlying model and data.

DIAGNOSTICS=ALL | NONE | *(keyword-list)*

DIAG=ALL | NONE | *(keyword-list)*

controls which diagnostics are produced. All the following diagnostics are produced when you specify DIAGNOSTICS=ALL. If you do not want any of these diagnostics, specify DIAGNOSTICS=NONE. If you want some but not all of the diagnostics, or if you want to change certain settings of these diagnostics, specify one or more of the following keywords. The default is DIAGNOSTICS=NONE.

AUTOCORR < (LAGS=*numeric-list***) >**

computes the autocorrelations at lags that are specified in the *numeric-list*. Elements in the *numeric-list* are truncated to integers, and repeated values are removed. If the LAGS= option is not specified, autocorrelations of lags 1, 5, and 10 are computed.

ESS

computes Carlin's estimate of the effective sample size, the correlation time, and the efficiency of the chain for each parameter.

GEWEKE < (*geweke-options***) >**

computes the Geweke spectral density diagnostics, which are essentially a two-sample t test between the first f_1 portion and the last f_2 portion of the chain. The defaults are $f_1 = 0.1$ and $f_2 = 0.5$, but you can choose other fractions by using the following *geweke-options*:

FRAC1=*value*

specifies the fraction f_1 for the first window.

FRAC2=*value*

specifies the fraction f_2 for the second window.

HEIDELBERGER < (*heidel-options***) >**

computes for each variable the Heidelberger and Welch diagnostic, which consists of a stationarity test of the null hypothesis that the sample values form a stationary process. If the stationarity test is not rejected, a halfwidth test is then carried out. Optionally, you can specify one or more of the following *heidel-options*:

EPS=*value*

specifies a positive number ϵ such that if the halfwidth is less than ϵ times the sample mean of the retained iterates, the halfwidth test is passed.

HALPHA=*value*

specifies the α level $(0 < \alpha < 1)$ for the halfwidth test.

SALPHA=*value*
specifies the α level ($0 < \alpha < 1$) for the stationarity test.

MCSE
MCERROR
computes the Monte Carlo standard error for each parameter. The Monte Carlo standard error, which measures the simulation accuracy, is the standard error of the posterior mean estimate and is calculated as the posterior standard deviation divided by the square root of the effective sample size.

RAFTERY< (*raftery-options***) >**
computes the Raftery and Lewis diagnostics, which evaluate the accuracy of the estimated quantile ($\hat{\theta}_Q$ for a given $Q \in (0,1)$) of a chain. $\hat{\theta}_Q$ can achieve any degree of accuracy when the chain is allowed to run for a long time. The computation stops when the estimated probability $\hat{P}_Q = \Pr(\theta \leq \hat{\theta}_Q)$ reaches within $\pm R$ of the value Q with probability S; that is, $\Pr(Q - R \leq \hat{P}_Q \leq Q + R) = S$. The following *raftery-options* enable you to specify Q, R, S, and a precision level ϵ for the test:

QUANTILE | Q=*value*
specifies the order (a value between 0 and 1) of the quantile of interest. The default is 0.025.

ACCURACY | R=*value*
specifies a small positive number as the margin of error for measuring the accuracy of the estimation of the quantile. The default is 0.005.

PROBABILITY | S=*value*
specifies the probability of attaining the accuracy of the estimation of the quantile. The default is 0.95.

EPSILON | EPS=*value*
specifies the tolerance level (a small positive number) for the stationary test. The default is 0.001.

MINTUNE=*number*
specifies the minimum number of tuning phases. The default is 2.

MAXTUNE=*number*
specifies the maximum number of tuning phases. The default is 24.

NBI=*number*
specifies the number of burn-in iterations before the chains are saved. The default is 1,000.

NMC=*number*
specifies the number of iterations after the burn-in. The default is 1,000.

NTU=*number*
specifies the number of samples for each tuning phase. The default is 500.

OUTPOST=*SAS-data-set*
names the SAS data set to contain the posterior samples. Alternatively, you can create the output data set by specifying an ODS OUTPUT statement as follows:

ODS OUTPUT POSTERIORSAMPLE *= < SAS-data-set>* **;**

PROPCOV=*value*

specifies the method that is used in constructing the initial covariance matrix for the Metropolis-Hastings algorithm. The QUANEW and NMSIMP methods find numerically approximated covariance matrices at the optimum of the posterior density function with respect to all continuous parameters. The tuning phase starts at the optimized values; in some problems, this can greatly increase convergence performance. If the approximated covariance matrix is not positive definite, then an identity matrix is used instead. You can specify the following *values*:

CONGRA

performs a conjugate-gradient optimization.

DBLDOG

performs a version of double-dogleg optimization.

NEWRAP

performs a Newton-Raphson optimization that combines a line-search algorithm with ridging.

NMSIMP

performs a Nelder-Mead simplex optimization.

NRRIDG

performs a Newton-Raphson optimization with ridging.

QUANEW

performs a quasi-Newton optimization.

TRUREG

performs a trust-region optimization.

SAMPLING=MULTIMETROPOLIS | UNIMETROPOLIS

specifies how to sample from the posterior distribution. SAMPLING=MULTIMETROPOLIS implements a Metropolis sampling scheme on a single block that contains all the parameters of the model. SAMPLING=UNIMETROPOLIS implements a Metropolis sampling scheme on multiple blocks, one for each parameter of the model. The default is SAMPLING=MULTIMETROPOLIS.

SEED=*number*

specifies an integer seed in the range 1 to $2^{31} - 1$ for the random number generator in the simulation. Specifying a seed enables you to reproduce identical Markov chains for the same specification. If you do not specify the SEED= option, or if you specify a nonpositive seed, a random seed is derived from the time of day.

STATISTICS < (*global-options***) > = ALL | NONE |** *keyword* **|** *(keyword-list)*

STATS < (*global-options***) > = ALL | NONE |** *keyword* **|** *(keyword-list)*

controls the number of posterior statistics that are produced. Specifying STATISTICS=ALL is equivalent to specifying STATISTICS=(CORR COV INTERVAL PRIOR SUMMARY). If you do not want any posterior statistics, specify STATISTICS=NONE. The default is STATISTICS=(SUMMARY INTERVAL). You can specify the following *global-options*:

ALPHA=*value* **<** *,value* **>. . . <** *,value* **>**

controls the probabilities of the credible intervals. The *value*, which must be between 0 and 1, produces a pair of 100(1–*value*)% equal-tail and highest posterior density (HPD) intervals for each parameter. The default is ALPHA=0.05, which yields the 95% credible intervals for each parameter.

PERCENT=*value* **<** *,value* **>. . . <** *,value* **>**

requests the percentile points of the posterior samples. The *value* must be between 0 and 100. The default is PERCENT=25, 50, 75, which yields the 25th, 50th, and 75th percentile points, respectively, for each parameter.

You can specify the following *keywords*:

CORR

produces the posterior correlation matrix.

COV

produces the posterior covariance matrix.

INTERVAL

produces equal-tail credible intervals and HPD intervals. The default is to produce the 95% equal-tail credible intervals and 95% HPD intervals, but you can use the ALPHA= *global-option* to request intervals of any probabilities.

NONE

suppresses printing of all summary statistics.

PRIOR

produces a summary table of the prior distributions that are used in the Bayesian analysis.

SUMMARY

produces the means, standard deviations, and percentile points (25th, 50th, and 75th) for the posterior samples. You can use the PERCENT= *global-option* to request specific percentile points.

THIN=*number*

THINNING=*number*

controls the thinning of the Markov chain. Only one in every k samples is used when THIN=k. If NBI=n_0 and NMC=n, the number of samples that are retained is

$$\left[\frac{n_0 + n}{k}\right] - \left[\frac{n_0}{k}\right]$$

where $[a]$ represents the integer part of the number a. The default is THIN=1.

BOUNDS Statement

BOUNDS *bound1* < *, bound2 . . .* > **;**

The BOUNDS statement imposes simple boundary constraints on the parameter estimates. BOUNDS statement constraints refer to the parameters that are estimated by the HPQLIM procedure. You can specify any number of BOUNDS statements.

Each *bound* is composed of parameters, constants, and inequality operators. Parameters that are associated with regressor variables are referred to by the names of the corresponding regressor variables. Specify each bound as follows:

item operator item < *operator item* < *operator item . . .* > >

Each *item* is a constant, the name of a parameter, or a list of parameter names. For more information about how parameters are named in the HPQLIM procedure, see the section "Naming of Parameters" on page 112. Each *operator* is <, >, <=, or >=.

You can use both the BOUNDS statement and the RESTRICT statement to impose boundary constraints; however, the BOUNDS statement provides a simpler syntax for specifying these types of constraints. For more information, see the section "RESTRICT Statement" on page 95.

The following BOUNDS statement constrains the estimates of the parameters that are associated with the variable ttime and the variables x1 through x10 to be between 0 and 1. The following example illustrates the use of parameter lists to specify boundary constraints.

```
bounds 0 < ttime x1-x10 < 1;
```

The following BOUNDS statement constrains the estimates of the correlation (_RHO) and sigma (_SIGMA) in the bivariate model:

```
bounds _rho >= 0, _sigma.y1 > 1, _sigma.y2 < 5;
```

ENDOGENOUS Statement

ENDOGENOUS *variables* ~ *options* **;**

The ENDOGENOUS statement specifies the type of dependent variables that appear on the left-hand side of the equation. The listed endogenous variables refer to the dependent variables that appear on the left-hand side of the equation. Currently, no right-hand side endogeneity is handled in PROC HPQLIM. All variables that appear on the right-hand side of the equation are treated as exogenous.

Censored Variable Options

CENSORED (*censored-options* **)**
specifies that the endogenous variables in this statement be censored. You can specify the following *censored-options*:

LB=*value* | *variable*

LOWERBOUND=*value* | *variable*

specifies the lower bound of the censored variables. If *value* is missing or the value in *variable* is missing, no lower bound is set. By default, no lower bound is set.

UB=*value* | *variable*

UPPERBOUND=*value* | *variable*

specifies the upper bound of the censored variables. If *value* is missing or the value in *variable* is missing, no upper bound is set. By default, no upper bound is set.

Truncated Variable Options

TRUNCATED (*truncated-options* **)**

You can specify the following *truncated-options*:

LB=*value* | *variable*

LOWERBOUND=*value* | *variable*

specifies the lower bound of the truncated variables. If *value* is missing or the value in *variable* is missing, no lower bound is set. By default, no lower bound is set.

UB=*value* | *variable*

UPPERBOUND=*value* | *variable*

specifies the upper bound of the truncated variables. If *value* is missing or the value in *variable* is missing, no upper bound is set. By default, no upper bound is set.

Stochastic Frontier Variable Options

FRONTIER < (*frontier-options* **) >**

You can specify the following *frontier-options*:

TYPE=HALF | EXPONENTIAL | TRUNCATED

specifies the model type.

HALF

specifies half-normal model.

EXPONENTIAL

specifies exponential model.

TRUNCATED

specifies truncated normal model.

PRODUCTION

specifies that the estimated model be a production function.

COST

specifies that the estimated model be a cost function.

If neither PRODUCTION nor COST is specified, a production function is estimated by default.

FREQ Statement

FREQ *variable* **;**

The FREQ statement identifies a variable that contains the frequency of occurrence of each observation. PROC HPQLIM treats each observation as if it appeared n times, where n is the value of the FREQ variable for the observation. If the frequency value is not an integer, it is truncated to an integer. If the frequency value is less than 1 or missing, the observation is not used in the model fitting. When the FREQ statement is not specified, each observation is assigned a frequency of 1. If you specify more than one FREQ statement, then the first FREQ statement is used.

HETERO Statement

HETERO *dependent variables* ~ *exogenous variables* < / *options* > **;**

The HETERO statement specifies variables that are related to the heteroscedasticity of the residuals and the way that these variables are used to model the error variance. PROC HPQLIM supports the following heteroscedastic regression model:

$$y_i = \mathrm{x}_i' \boldsymbol{\beta} + \epsilon_i$$

$$\epsilon_i \sim \mathrm{N}(0, \sigma_i^2)$$

For more information about the specification of functional forms, see the section "Heteroscedasticity" on page 101. The following *options* specify the functional forms of heteroscedasticity:

LINK=EXP | LINEAR

specifies the functional form.

EXP

specifies the exponential link function:

$$\sigma_i^2 \quad = \quad \sigma^2(1 + \exp(\mathbf{z}_i' \boldsymbol{\gamma}))$$

LINEAR

specifies the linear link function:

$$\sigma_i^2 \quad = \quad \sigma^2(1 + \mathbf{z}_i' \boldsymbol{\gamma})$$

The default is LINK=EXP.

NOCONST

specifies that there be no constant in the linear or exponential heteroscedasticity model:

$$\begin{aligned} \sigma_i^2 &= \sigma^2(\mathbf{z}_i' \boldsymbol{\gamma}) \\ \sigma_i^2 &= \sigma^2 \exp(\mathbf{z}_i' \boldsymbol{\gamma}) \end{aligned}$$

This option is ignored if you do not specify the LINK= option.

SQUARE

estimates the model by using the square of the linear heteroscedasticity function. For example, you can specify the following heteroscedasticity function:

$$\sigma_i^2 = \sigma^2(1 + (\mathbf{z}_i^{'}\boldsymbol{\gamma})^2)$$

```
model y = x1 x2 / censored(lb=0);
hetero y ~ z1 / link=linear square;
```

The SQUARE option does not apply to the exponential heteroscedasticity function because the square of an exponential function of $\mathbf{z}_i^{'}\boldsymbol{\gamma}$ is the same as the exponential of $2\mathbf{z}_i^{'}\boldsymbol{\gamma}$. Hence, the only difference is that all $\boldsymbol{\gamma}$ estimates are divided by two.

This option is ignored if you do not specify the LINK= option. You cannot use the HETERO statement within a Bayesian framework.

INIT Statement

INIT *initvalue1* < , *initvalue2* . . . > ;

The INIT statement sets initial values for parameters in the optimization. You can specify any number of INIT statements.

Each *initvalue* is written as a parameter or parameter list, followed by an optional equality operator (=), followed by a number:

parameter **<=>** *number*

MODEL Statement

MODEL *dependent* **=** *regressors* < */ options* > ;

The MODEL statement specifies the dependent variable and independent regressor variables for the regression model.

You can specify the following *option* after a slash (/).

NOINT

suppresses the intercept parameter.

You can also specify the following endogenous variable options, which are the same as the options that are specified in the ENDOGENOUS statement. If an endogenous variable option is specified in both the MODEL statement and the ENDOGENOUS statement, the option in the ENDOGENOUS statement is used.

Censored Variable Options

CENSORED < (*censored-options*) >

specifies that the endogenous variables in this statement be censored. You can specify the following *censored-options*:

LB=*value* | *variable*

LOWERBOUND=*value* | *variable*

specifies the lower bound of the censored variables. If *value* is missing or the value in *variable* is missing, no lower bound is set. By default, no lower bound is set.

UB=*value* | *variable*

UPPERBOUND=*value* | *variable*

specifies the upper bound of the censored variables. If *value* is missing or the value in *variable* is missing, no upper bound is set. By default, no upper bound is set.

Truncated Variable Options

TRUNCATED < (*truncated-options*) >

You can specify the following *truncated-options*:

LB=*value* | *variable*

LOWERBOUND=*value* | *variable*

specifies the lower bound of the truncated variables. If *value* is missing or the value in *variable* is missing, no lower bound is set. By default, no lower bound is set.

UB=*value* | *variable*

UPPERBOUND=*value* | *variable*

specifies the upper bound of the truncated variables. If *value* is missing or the value in *variable* is missing, no upper bound is set. By default, no upper bound is set.

Stochastic Frontier Variable Options

FRONTIER < (*frontier-options*) >

You can specify the following *frontier-options*:

TYPE=HALF | EXPONENTIAL | TRUNCATED

specifies the model type.

HALF

specifies a half-normal model.

EXPONENTIAL

specifies an exponential model.

TRUNCATED

specifies a truncated normal model.

PRODUCTION
: specifies that the estimated model be a production function.

COST
: specifies that the estimated model be a cost function.

If neither PRODUCTION nor COST is specified, a production function is estimated by default.

OUTPUT Statement

OUTPUT OUT=*SAS-data-set* < *output-options* > **;**

The OUTPUT statement creates a new SAS data set to contain variables that are specified with the COPYVAR option and the following data if they are specified by *output-options*: estimates of $\mathbf{x}'\boldsymbol{\beta}$, predicted value, residual, marginal effects, probability, standard deviation of the error, expected value, conditional expected value, technical efficiency measures, and inverse Mills ratio. When the response values are missing for the observation, all output estimates except the residual are still computed as long as none of the explanatory variables are missing. This enables you to compute these statistics for prediction. You can specify only one OUTPUT statement.

You must specify the OUT= option:

OUT=*SAS-data-set*
: names the output data set.

COPYVAR=*SAS-variable-names*
COPYVARS=*SAS-variable-names*
: adds SAS variables to the output data set

You can specify one or more of the following *output-options*:

CONDITIONAL
: outputs estimates of conditional expected values of continuous endogenous variables.

ERRSTD
: outputs estimates of σ_j, the standard deviation of the error term.

EXPECTED
: outputs estimates of expected values of continuous endogenous variables.

MARGINAL
: outputs marginal effects.

OUTVAR(*SAS-variable-names***)**
: adds SAS variables to the output data set.

PREDICTED
: outputs estimates of predicted endogenous variables.

RESIDUAL
outputs estimates of residuals of continuous endogenous variables.

XBETA
outputs estimates of $\mathbf{x}'\boldsymbol{\beta}$.

TE1
outputs estimates of technical efficiency for each producer in the stochastic frontier model that is suggested by Battese and Coelli (1988).

TE2
outputs estimates of technical efficiency for each producer in the stochastic frontier model that is suggested by Jondrow et al. (1982).

PERFORMANCE Statement

PERFORMANCE < *performance-options* > **;**

The PERFORMANCE statement specifies *performance-options* to control the multithreaded and distributed computing environment and requests detailed performance results of the HPQLIM procedure. You can also use the PERFORMANCE statement to control whether the HPQLIM procedure executes in single-machine or distributed mode. You can specify the following *performance-options*:

DETAILS
requests a table that shows a timing breakdown of the procedure steps.

NODES=*n*
specifies the number of nodes in the distributed computing environment, provided that the data are not processed alongside the database.

NTHREADS=*n*
specifies the number of threads for analytic computations and overrides the SAS system option THREADS | NOTHREADS. If you do not specify the NTHREADS= option, PROC HPQLIM creates one thread per CPU for the analytic computations.

The PERFORMANCE statement is documented further in the section "PERFORMANCE Statement" on page 34 in Chapter 2, "Shared Concepts and Topics."

PRIOR Statement

PRIOR _REGRESSORS | *parameter-list* ~ *distribution* **;**

The PRIOR statement specifies the prior distribution of the model parameters. You must specify one parameter or a list of parameters, a tilde ~, and then a distribution with its parameters. Multiple PRIOR statements are allowed.

You can specify the following *distributions*:

NORMAL(MEAN=μ, VAR=σ^2)
specifies a normal distribution with the parameters MEAN and VAR.

GAMMA(SHAPE=a, SCALE=b)
specifies a gamma distribution with the parameters SHAPE and SCALE.

IGAMMA(SHAPE=a, SCALE=b)
specifies an inverse gamma distribution with the parameters SHAPE and SCALE.

UNIFORM(MIN=m, MAX=M)
specifies a uniform distribution that is defined between MIN and MAX.

BETA(SHAPE1=a, SHAPE2=b, MIN=m, MAX=M)
specifies a beta distribution with the parameters SHAPE1 and SHAPE2 and defined between MIN and MAX.

T(LOCATION=μ, DF=ν)
specifies a noncentral t distribution with DF degrees of freedom and a location parameter equal to LOCATION.

For more information about how to specify *distributions*, see the section "Standard Distributions" on page 106.

You can specify the special keyword REGRESSORS to select all the parameters that are used in the linear regression component of the model.

RESTRICT Statement

RESTRICT *restriction1 < , restriction2 . . . >* **;**

The RESTRICT statement imposes linear restrictions on the parameter estimates. You can specify any number of RESTRICT statements, but the number of restrictions that are imposed is limited by the number of regressors.

Each *restriction* is written as an expression, followed by an equality operator (=) or an inequality operator (<, >, <=, >=), followed by a second expression:

expression operator expression

The *operator* can be =, <, >, <= , or >=. The *operator* and second *expression* are optional.

Restriction expressions can be composed of parameter names; multiplication ($*$), addition ($+$), and substitution ($-$) operators; and constants. Parameters that are named in restriction expressions must be among the parameters that are estimated by the model. Parameters that are associated with a regressor variable are referred to by the name of the corresponding regressor variable. The restriction expressions must be a linear function of the parameters.

The following statements illustrate the use of the RESTRICT statement:

```
proc hpqlim data=one;
   model y = x1-x10 / censored(lb=0);
   restrict x1*2 <= x2 + x3;
run;
```

TEST Statement

<'label':> **TEST** *<'string':> equation < ,equation. . . > / options* **;**

The TEST statement performs Wald, Lagrange multiplier, and likelihood ratio tests of linear hypotheses about the regression parameters in the preceding MODEL statement. Each equation specifies a linear hypothesis to be tested. All hypotheses in one TEST statement are tested jointly. Variable names in the equations must correspond to regressors in the preceding MODEL statement, and each name represents the coefficient of the corresponding regressor. Use the keyword INTERCEPT for a test that includes a constant.

You can specify the following *options* after the slash (/):

ALL

requests Wald, Lagrange multiplier, and likelihood ratio tests.

LM

requests the Lagrange multiplier test.

LR

requests the likelihood ratio test.

WALD

requests the Wald test.

The following statements illustrate the use of the TEST statement (note the use of the INTERCEPT keyword in the second TEST statement):

```
proc hpqlim;
   model y = x1 x2 x3;
   test x1 = 0, x2 * .5 + 2 * x3 = 0;
   test _int: test intercept = 0, x3 = 0;
run;
```

The first TEST statement investigates the joint hypothesis that

$$\beta_1 = 0$$

and

$$0.5\beta_2 + 2\beta_3 = 0$$

Only linear equality restrictions and tests are permitted in PROC HPQLIM. Test expressions can be composed only of algebraic operations that involve the addition symbol (+), subtraction symbol (–), and multiplication symbol (*).

The TEST statement accepts labels that are reproduced in the printed output. You can label a TEST statement in two ways: you can specify a label followed by a colon before the TEST keyword, or you can specify a quoted string after the TEST keyword. If you specify both a label before the TEST keyword and a quoted string after the keyword, PROC HPQLIM uses the label that precedes the colon. If no label or quoted string is specified, PROC HPQLIM labels the test automatically.

WEIGHT Statement

WEIGHT *variable* < / *option*> **;**

The WEIGHT statement specifies a variable that supplies weighting values to use for each observation in estimating parameters. The log likelihood for each observation is multiplied by the corresponding weight variable value.

If the weight of an observation is nonpositive, that observation is not used in the estimation.

You can add the following *option* after a slash (/):

NONORMALIZE
specifies that the weights must be used as is. When this option is not specified, the weights are normalized so that they add up to the actual sample size. Weights w_i are normalized by multiplying them by $\frac{n}{\sum_{i=1}^{n} w_i}$, where n is the sample size.

Details: HPQLIM Procedure

Limited Dependent Variable Models

Censored Regression Models

When the dependent variable is censored, values in a certain range are all transformed to a single value. For example, the standard Tobit model can be defined as

$$y_i^* = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

$$y_i = \begin{cases} y_i^* & \text{if}\, y_i^* > 0 \\ 0 & \text{if}\, y_i^* \leq 0 \end{cases}$$

where $\epsilon_i \sim iidN(0, \sigma^2)$. The log-likelihood function of the standard censored regression model is

$$\ell = \sum_{i \in \{y_i = 0\}} \ln[1 - \Phi(\mathbf{x}_i'\boldsymbol{\beta}/\sigma)] + \sum_{i \in \{y_i > 0\}} \ln\left[\phi(\frac{y_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma})/\sigma\right]$$

where $\Phi(\cdot)$ is the cumulative density function of the standard normal distribution and $\phi(\cdot)$ is the probability density function of the standard normal distribution.

The Tobit model can be generalized to handle observation-by-observation censoring. The censored model on both the lower and upper limits can be defined as

$$y_i = \begin{cases} R_i & \text{if } y_i^* \geq R_i \\ y_i^* & \text{if } L_i < y_i^* < R_i \\ L_i & \text{if } y_i^* \leq L_i \end{cases}$$

The log-likelihood function can be written as

$$\ell \; = \sum_{i \in \{L_i < y_i < R_i\}} \ln\left[\phi(\frac{y_i - \mathrm{x}_i'\boldsymbol{\beta}}{\sigma})/\sigma\right] + \sum_{i \in \{y_i = R_i\}} \ln\left[\Phi(-\frac{R_i - \mathrm{x}_i'\boldsymbol{\beta}}{\sigma})\right] + \sum_{i \in \{y_i = L_i\}} \ln\left[\Phi(\frac{L_i - \mathrm{x}_i'\boldsymbol{\beta}}{\sigma})\right]$$

Log-likelihood functions of the lower-limit or upper-limit censored model are easily derived from the two-limit censored model. The log-likelihood function of the lower-limit censored model is

$$\ell = \sum_{i \in \{y_i > L_i\}} \ln\left[\phi(\frac{y_i - \mathrm{x}_i'\boldsymbol{\beta}}{\sigma})/\sigma\right] + \sum_{i \in \{y_i = L_i\}} \ln\left[\Phi(\frac{L_i - \mathrm{x}_i'\boldsymbol{\beta}}{\sigma})\right]$$

The log-likelihood function of the upper-limit censored model is

$$\ell = \sum_{i \in \{y_i < R_i\}} \ln\left[\phi(\frac{y_i - \mathrm{x}_i'\boldsymbol{\beta}}{\sigma})/\sigma\right] + \sum_{i \in \{y_i = R_i\}} \ln\left[1 - \Phi(\frac{R_i - \mathrm{x}_i'\boldsymbol{\beta}}{\sigma})\right]$$

Truncated Regression Models

In a truncated model, the observed sample is a subset of the population where the dependent variable falls within a certain range. For example, when neither a dependent variable nor exogenous variables are observed for $y_i^* \le 0$, the truncated regression model can be specified as

$$\ell = \sum_{i \in \{y_i > 0\}} \left\{-\ln\Phi(\mathrm{x}_i'\boldsymbol{\beta}/\sigma) + \ln\left[\frac{\phi((y_i - \mathrm{x}_i'\boldsymbol{\beta})/\sigma)}{\sigma}\right]\right\}$$

The two-limit truncation model is defined as

$$y_i = y_i^* \text{if } L_i < y_i^* < R_i$$

The log-likelihood function of the two-limit truncated regression model is

$$\ell = \sum_{i=1}^{N} \left\{\ln\left[\phi(\frac{y_i - \mathrm{x}_i'\boldsymbol{\beta}}{\sigma})/\sigma\right] - \ln\left[\Phi(\frac{R_i - \mathrm{x}_i'\boldsymbol{\beta}}{\sigma}) - \Phi(\frac{L_i - \mathrm{x}_i'\boldsymbol{\beta}}{\sigma})\right]\right\}$$

The log-likelihood function of the lower-limit truncation model is

$$\ell = \sum_{i=1}^{N} \left\{\ln\left[\phi(\frac{y_i - \mathrm{x}_i'\boldsymbol{\beta}}{\sigma})/\sigma\right] - \ln\left[1 - \Phi(\frac{L_i - \mathrm{x}_i'\boldsymbol{\beta}}{\sigma})\right]\right\}$$

The log-likelihood function of the upper-limit truncation model is

$$\ell = \sum_{i=1}^{N} \left\{\ln\left[\phi(\frac{y_i - \mathrm{x}_i'\boldsymbol{\beta}}{\sigma})/\sigma\right] - \ln\left[\Phi(\frac{R_i - \mathrm{x}_i'\boldsymbol{\beta}}{\sigma})\right]\right\}$$

Stochastic Frontier Production and Cost Models

Stochastic frontier production models were first developed by Aigner, Lovell, and Schmidt (1977); Meeusen and van den Broeck (1977). Specification of these models allow for random shocks of the production or cost but also include a term for technical or cost inefficiency. Assuming that the production function takes a log-linear Cobb-Douglas form, the stochastic frontier production model can be written as

$$ln(y_i) = \beta_0 + \sum_n \boldsymbol{\beta}_n \ln(x_{ni}) + \epsilon_i$$

where $\epsilon_i = v_i - u_i$. The v_i term represents the stochastic error component, and the u_i term represents the nonnegative, technical inefficiency error component. The v_i error component is assumed to be distributed iid normal and independent from u_i. If $u_i > 0$, the error term ϵ_i is negatively skewed and represents technical inefficiency. If $u_i < 0$, the error term ϵ_i is positively skewed and represents cost inefficiency. PROC HPQLIM models the u_i error component as a half-normal, exponential, or truncated normal distribution.

The Normal-Half-Normal Model

When v_i is iid $N(0, \sigma_v^2)$ in a normal-half-normal model, u_i is iid $N^+(0, \sigma_u^2)$, with v_i and u_i independent of each other. Given the independence of error terms, the joint density of v and u can be written as

$$f(u, v) = \frac{2}{2\pi\sigma_u\sigma_v} \exp\left\{-\frac{u^2}{2\sigma_u^2} - \frac{v^2}{2\sigma_v^2}\right\}$$

Substituting $v = \epsilon + u$ into the preceding equation gives

$$f(u, \epsilon) = \frac{2}{2\pi\sigma_u\sigma_v} \exp\left\{-\frac{u^2}{2\sigma_u^2} - \frac{(\epsilon + u)^2}{2\sigma_v^2}\right\}$$

Integrating u out to obtain the marginal density function of ϵ results in the following form:

$$\begin{aligned} f(\epsilon) &= \int_0^\infty f(u, \epsilon)du \\ &= \frac{2}{\sqrt{2\pi}\sigma}\left[1 - \phi\left(\frac{\epsilon\lambda}{\sigma}\right)\right] \exp\left\{-\frac{\epsilon^2}{2\sigma^2}\right\} \\ &= \frac{2}{\sigma}\phi\left(\frac{\epsilon}{\sigma}\right)\Phi\left(-\frac{\epsilon\lambda}{\sigma}\right) \end{aligned}$$

where $\lambda = \sigma_u/\sigma_v$ and $\sigma = \sqrt{\sigma_u^2 + \sigma_v^2}$.

In the case of a stochastic frontier cost model, $v = \epsilon - u$ and

$$f(\epsilon) = \frac{2}{\sigma}\phi\left(\frac{\epsilon}{\sigma}\right)\Phi\left(\frac{\epsilon\lambda}{\sigma}\right)$$

The log-likelihood function for the production model with N producers is written as

$$\ln L = \text{constant} - N\ln\sigma + \sum_i \ln\Phi\left(-\frac{\epsilon_i\lambda}{\sigma}\right) - \frac{1}{2\sigma^2}\sum_i \epsilon_i^2$$

The Normal-Exponential Model

Under the normal-exponential model, v_i is iid $N(0, \sigma_v^2)$ and u_i is iid exponential. Given the independence of error term components u_i and v_i, the joint density of v and u can be written as

$$f(u, v) = \frac{1}{\sqrt{2\pi}\sigma_u \sigma_v} \exp\left\{-\frac{u}{\sigma_u} - \frac{v^2}{2\sigma_v^2}\right\}$$

The marginal density function of ϵ for the production function is

$$\begin{aligned} f(\epsilon) &= \int_0^\infty f(u, \epsilon) du \\ &= \left(\frac{1}{\sigma_u}\right) \Phi\left(-\frac{\epsilon}{\sigma_v} - \frac{\sigma_v}{\sigma_u}\right) \exp\left\{\frac{\epsilon}{\sigma_u} + \frac{\sigma_v^2}{2\sigma_u^2}\right\} \end{aligned}$$

The marginal density function for the cost function is equal to

$$f(\epsilon) = \left(\frac{1}{\sigma_u}\right) \Phi\left(\frac{\epsilon}{\sigma_v} - \frac{\sigma_v}{\sigma_u}\right) \exp\left\{-\frac{\epsilon}{\sigma_u} + \frac{\sigma_v^2}{2\sigma_u^2}\right\}$$

The log-likelihood function for the normal-exponential production model with N producers is

$$\ln L = \text{constant} - N \ln \sigma_u + N \left(\frac{\sigma_v^2}{2\sigma_u^2}\right) + \sum_i \frac{\epsilon_i}{\sigma_u} + \sum_i \ln \Phi\left(\frac{\epsilon_i}{\sigma_v} - \frac{\sigma_v}{\sigma_u}\right)$$

The Normal–Truncated Normal Model

The normal–truncated normal model is a generalization of the normal-half-normal model that allows the mean of u_i to differ from zero. Under the normal–truncated normal model, the error term component v_i is iid $N^+(0, \sigma_v^2)$ and u_i is iid $N(\mu, \sigma_u^2)$. The joint density of v_i and u_i can be written as

$$f(u, v) = \frac{1}{\sqrt{2\pi}\sigma_u \sigma_v \Phi(\mu/\sigma_u)} \exp\left\{-\frac{(u-\mu)^2}{2\sigma_u^2} - \frac{v^2}{2\sigma_v^2}\right\}$$

The marginal density function of ϵ for the production function is

$$\begin{aligned} f(\epsilon) &= \int_0^\infty f(u, \epsilon) du \\ &= \frac{1}{\sqrt{2\pi}\sigma \Phi(\mu/\sigma_u)} \Phi\left(\frac{\mu}{\sigma\lambda} - \frac{\epsilon\lambda}{\sigma}\right) \exp\left\{-\frac{(\epsilon+\mu)^2}{2\sigma^2}\right\} \\ &= \frac{1}{\sigma}\phi\left(\frac{\epsilon+\mu}{\sigma}\right) \Phi\left(\frac{\mu}{\sigma\lambda} - \frac{\epsilon\lambda}{\sigma}\right) \left[\Phi\left(\frac{\mu}{\sigma_u}\right)\right]^{-1} \end{aligned}$$

The marginal density function for the cost function is

$$f(\epsilon) = \frac{1}{\sigma}\phi\left(\frac{\epsilon-\mu}{\sigma}\right) \Phi\left(\frac{\mu}{\sigma\lambda} + \frac{\epsilon\lambda}{\sigma}\right) \left[\Phi\left(\frac{\mu}{\sigma_u}\right)\right]^{-1}$$

The log-likelihood function for the normal–truncated normal production model with N producers is

$$\begin{aligned}\ln L &= \text{constant} - N\ln\sigma - N\ln\Phi\left(\frac{\mu}{\sigma_u}\right) + \sum_i \ln\Phi\left(\frac{\mu}{\sigma\lambda} + \frac{\epsilon_i\lambda}{\sigma}\right)\\ &\quad -\frac{1}{2}\sum_i\left(\frac{\epsilon_i+\mu}{\sigma}\right)^2\end{aligned}$$

For more information about normal-half-normal, normal-exponential, and normal–truncated normal models, see: Kumbhakar and Lovell (2000); Coelli, Prasada Rao, and Battese (1998).

Heteroscedasticity

If the variance of regression disturbance, (ϵ_i), is heteroscedastic, the variance can be specified as a function of variables

$$E(\epsilon_i^2) = \sigma_i^2 = f(\mathbf{z}_i'\boldsymbol{\gamma})$$

Table 4.2 shows various functional forms of heteroscedasticity and the corresponding options to request each model.

Table 4.2 Specification Summary for Modeling Heteroscedasticity

Number	Model	Options
1	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(1+\exp(\mathbf{z}_i'\gamma))$	LINK=EXP (default)
2	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2\exp(\mathbf{z}_i'\gamma)$	LINK=EXP NOCONST
3	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(1+\sum_{l=1}^{L}\gamma_l z_{li})$	LINK=LINEAR
4	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(1+(\sum_{l=1}^{L}\gamma_l z_{li})^2)$	LINK=LINEAR SQUARE
5	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(\sum_{l=1}^{L}\gamma_l z_{li})$	LINK=LINEAR NOCONST
6	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2((\sum_{l=1}^{L}\gamma_l z_{li})^2)$	LINK=LINEAR SQUARE NOCONST

In models 3 and 5, variances of some observations might be negative. Although the HPQLIM procedure assigns a large penalty to move the optimization away from such a region, the optimization might not be able to improve the objective function value and might become locked in the region. Signs of such an outcome include extremely small likelihood values or missing standard errors in the estimates. In models 2 and 6, variances are guaranteed to be greater than or equal to zero, but variances of some observations might be very close to 0. In these scenarios, standard errors might be missing. Models 1 and 4 do not have such problems. Variances in these models are always positive and never close to 0.

The heteroscedastic regression model is estimated using the following log-likelihood function, where $e_i = y_i - \mathbf{x}_i'\boldsymbol{\beta}$:

$$\ell = -\frac{N}{2}\ln(2\pi) - \sum_{i=1}^{N}\frac{1}{2}\ln(\sigma_i^2) - \frac{1}{2}\sum_{i=1}^{N}(\frac{e_i}{\sigma_i})^2$$

Tests on Parameters

In general, the tested hypothesis can be written as

$$H_0 : \mathbf{h}(\theta) = 0$$

where $\mathbf{h}(\theta)$ is an $r \times 1$ vector-valued function of the parameters θ given by the r expressions that are specified in the TEST statement.

Let $\hat{V}$ be the estimate of the covariance matrix of $\hat{\theta}$. Let $\hat{\theta}$ be the unconstrained estimate of θ and $\tilde{\theta}$ be the constrained estimate of θ such that $h(\tilde{\theta}) = 0$. Let

$$A(\theta) = \partial h(\theta)/\partial \theta \mid_{\hat{\theta}}$$

Using this notation, the test statistics for the three types of tests are computed as follows.

- The Wald test statistic is defined as

  $$W = h^{'}(\hat{\theta}) \left(A(\hat{\theta}) \hat{V} A^{'}(\hat{\theta}) \right)^{-1} h(\hat{\theta})$$

  The Wald test is not invariant to reparameterization of the model (Gregory and Veall 1985; Gallant 1987, p. 219). For more information about the theoretical properties of the Wald test, see Phillips and Park (1988).

- The Lagrange multiplier test statistic is

  $$LM = \lambda^{'} A(\tilde{\theta}) \tilde{V} A^{'}(\tilde{\theta}) \lambda$$

  where λ is the vector of Lagrange multipliers from the computation of the restricted estimate $\tilde{\theta}$.

- The likelihood ratio test statistic is

  $$LR = 2 \left(L(\hat{\theta}) - L(\tilde{\theta}) \right)$$

  where $\tilde{\theta}$ represents the constrained estimate of θ and L is the concentrated log-likelihood value.

For each type of test, under the null hypothesis the test statistic is asymptotically distributed as a χ^2 random variable with r degrees of freedom, where r is the number of expressions in the TEST statement. The p-values reported for the tests are computed from the $\chi^2(r)$ distribution and are only asymptotically valid.

Monte Carlo simulations suggest that the asymptotic distribution of the Wald test is a poorer approximation to its small sample distribution than that of the other two tests. However, the Wald test has the lowest computational cost, because it does not require computation of the constrained estimate $\tilde{\theta}$.

The following statements use the TEST statement to perform a likelihood ratio test:

```
proc hpqlim;
   model y = x1 x2 x3;
   test x1 = 0, x2 * .5 + 2 * x3 = 0 /lr;
run;
```

Bayesian Analysis

To perform Bayesian analysis, you must specify a BAYES statement. Unless otherwise stated, all options that are described in this section are options in the BAYES statement.

By default, PROC HPQLIM uses the random walk Metropolis algorithm to obtain posterior samples. For the implementation details of the Metropolis algorithm in PROC HPQLIM, such as the blocking of the parameters and tuning of the covariance matrices, see the sections "Blocking of Parameters" on page 103 and "Tuning the Proposal Distribution" on page 103.

The Bayes theorem states that

$$p(\theta|\mathbf{y}) \propto \pi(\theta)L(y|\theta)$$

where θ is a parameter or a vector of parameters and $\pi(\theta)$ is the product of the prior densities that are specified in the PRIOR statement. The term $L(y|\theta)$ is the likelihood that is associated with the MODEL statement.

Blocking of Parameters

In a multivariate parameter model, all the parameters are updated in one single block (by default or when you specify the SAMPLING=MULTIMETROPOLIS option). This can be inefficient, especially when parameters have vastly different scales. As an alternative, you can update the parameters one at a time (by specifying SAMPLING=UNIMETROPOLIS).

Tuning the Proposal Distribution

One key factor in achieving high efficiency of a Metropolis-based Markov chain is finding a good proposal distribution for each block of parameters. This process is called tuning. The tuning phase consists of a number of loops that are controlled by the options MINTUNE and MAXTUNE. The MINTUNE= option controls the minimum number of tuning loops and has a default value of 2. The MAXTUNE= option controls the maximum number of tuning loops and has a default value of 24. Each loop repeats the number of times specified by the NTU= option, which has a default of 500. At the end of every loop, PROC HPQLIM examines the acceptance probability for each block. The acceptance probability is the percentage of NTU proposed values that have been accepted. If this probability does not fall within the acceptance tolerance range (see the following section), the proposal distribution is modified before the next tuning loop.

A good proposal distribution should resemble the actual posterior distribution of the parameters. Large sample theory states that the posterior distribution of the parameters approaches a multivariate normal distribution (Gelman et al. 2004, Appendix B; Schervish 1995, Section 7.4). That is why a normal proposal distribution often works well in practice. The default proposal distribution in PROC HPQLIM is the normal distribution.

Scale Tuning

The acceptance rate is closely related to the sampling efficiency of a Metropolis chain. For a random walk Metropolis, a high acceptance rate means that most new samples occur right around the current data point. Their frequent acceptance means that the Markov chain is moving rather slowly and not exploring the parameter space fully. A low acceptance rate means that the proposed samples are often rejected; hence the chain is not moving much. An efficient Metropolis sampler has an acceptance rate that is neither too high nor too low. The scale c in the proposal distribution $q(\cdot|\cdot)$ effectively controls this acceptance probability.

Roberts, Gelman, and Gilks (1997) show that if both the target densities and the proposal densities are normal, the optimal acceptance probability for the Markov chain should be around 0.45 in a one-dimension problem and should asymptotically approach 0.234 in higher-dimension problems. The corresponding optimal scale is 2.38, which is the initial scale that is set for each block.

Because of the nature of stochastic simulations, it is impossible to fine-tune a set of variables so that the Metropolis chain has exactly the desired acceptance rate that you want. In addition, Roberts and Rosenthal (2001) empirically demonstrate that an acceptance rate between 0.15 and 0.5 is at least 80% efficient, so there is really no need to fine-tune the algorithms to reach an acceptance probability that is within a small tolerance of the optimal values. PROC HPQLIM works with a probability range, determined by a target acceptance ± 0.075. If the observed acceptance rate in a given tuning loop is less than the lower bound of the range, the scale is reduced; if the observed acceptance rate is greater than the upper bound of the range, the scale is increased. During the tuning phase, a scale parameter in the normal distribution is adjusted as a function of the observed acceptance rate and the target acceptance rate. PROC HPQLIM uses the updating scheme[1]

$$c_{\text{new}} = \frac{c_{\text{cur}} \cdot \Phi^{-1}(p_{\text{opt}}/2)}{\Phi^{-1}(p_{\text{cur}}/2)}$$

where c_{cur} is the current scale, p_{cur} is the current acceptance rate, and p_{opt} is the optimal acceptance probability.

Covariance Tuning

To tune a covariance matrix, PROC HPQLIM takes a weighted average of the old proposal covariance matrix and the recent observed covariance matrix, based on the number of samples (as specified by the NTU= option) in the current loop. The formula to update the covariance matrix is

$$\text{COV}_{\text{new}} = 0.75\ \text{COV}_{\text{cur}} + 0.25\ \text{COV}_{\text{old}}$$

There are two ways to initialize the covariance matrix:

- The default is an identity matrix that is multiplied by the initial scale of 2.38 and divided by the square root of the number of estimated parameters in the model. A number of tuning phases might be required before the proposal distribution is tuned to its optimal stage, because the Markov chain needs to spend time to learn about the posterior covariance structure. If the posterior variances of your parameters vary by more than a few orders of magnitude, if the variances of your parameters are much different from 1, or if the posterior correlations are high, then the proposal tuning algorithm might have difficulty forming an acceptable proposal distribution.
- Alternatively, you can use a numerical optimization routine, such as the quasi-Newton method, to find a starting covariance matrix. The optimization is performed on the joint posterior distribution, and the covariance matrix is a quadratic approximation at the posterior mode. In some cases this is a better and more efficient way of initializing the covariance matrix. However, there are cases, such as when the number of parameters is large, where the optimization could fail to find a matrix that is positive definite. In those cases, the tuning covariance matrix is reset to the identity matrix.

[1] Roberts, Gelman, and Gilks (1997) and Roberts and Rosenthal (2001) demonstrate that the relationship between acceptance probability and scale in a random walk Metropolis scheme is $p = 2\Phi\left(-\sqrt{I}c/2\right)$, where c is the scale, p is the acceptance rate, Φ is the CDF of a standard normal, and $I \equiv E_f[(f'(x)/f(x))^2]$, where $f(x)$ is the density function of samples. This relationship determines the updating scheme, with I replaced by the identity matrix to simplify calculation.

A by-product of the optimization routine is that it also finds the maximum a posteriori (MAP) estimates with respect to the posterior distribution. The MAP estimates are used as the initial values of the Markov chain.

For more information, see the section "INIT Statement" on page 91.

Initial Values of the Markov Chains

You can assign initial values to any parameters. For more information, see the INIT statement. If you use the optimization PROPCOV= option, PROC HPQLIM starts the tuning at the optimized values. This option overwrites the provided initial values.

Prior Distributions

The PRIOR statement specifies the prior distribution of the model parameters. You must specify one parameter or a list of parameters, a tilde (~), and then a distribution with its parameters. You can specify multiple PRIOR statements to define independent priors. Parameters that are associated with a regressor variable are referred to by the name of the corresponding regressor variable.

You can specify the special keyword _REGRESSORS to consider all the regressors of a model. If multiple PRIOR statements affect the same parameter, the last PRIOR statement prevails. For example, in a regression with two regressors (X1, X2), the following statements imply that the prior on X1 is NORMAL(MEAN=0, VAR=1), the prior on X2 is GAMMA(SHAPE=3, SCALE=4).

```
...
prior _Regressors ~ uniform(min=0, max=1);
prior X1 X2 ~ gamma(shape=3, scale=4);
prior X1 ~ normal(mean=0, var=1);
...
```

If a parameter is not associated with a PRIOR statement or if some of the prior hyperparameters are missing, then the default choices in Table Table 4.3 are considered.

Table 4.3 Default Values for Prior Distributions

PRIOR Distribution	Hyperparameter$_1$	Hyperparameter$_2$	Min	Max	Parameters Default Choice
NORMAL	MEAN=0	VAR=1E6	$-\infty$	∞	Regression-Location-Threshold
IGAMMA	SHAPE=2.000001	SCALE=1	> 0	∞	Scale
GAMMA	SHAPE=1	SCALE=1	0	∞	
UNIFORM			$-\infty$	∞	
BETA	SHAPE1=1	SHAPE2=1	$-\infty$	∞	
T	LOCATION=0	DF=3	$-\infty$	∞	

For density specification, see the section "Standard Distributions" on page 106.

Standard Distributions

Table 4.4 through Table 4.9 show all the distribution density functions that PROC HPQLIM recognizes. You specify these distribution densities in the PRIOR statement.

Table 4.4 Beta Distribution

PRIOR statement	BETA(SHAPE1=a, SHAPE2=b, MIN=m, MAX=M)
	Note: Commonly $m = 0$ and $M = 1$.
Density	$\frac{(\theta-m)^{a-1}(M-\theta)^{b-1}}{B(a,b)(M-m)^{a+b-1}}$
Parameter restriction	$a > 0, \quad b > 0, \quad -\infty < m < M < \infty$
Range	$\begin{cases} [m, M] & \text{when } a = 1, b = 1 \\ [m, M) & \text{when } a = 1, b \neq 1 \\ (m, M] & \text{when } a \neq 1, b = 1 \\ (m, M) & \text{otherwise} \end{cases}$
Mean	$\frac{a}{a+b} \times (M - m) + m$
Variance	$\frac{ab}{(a+b)^2(a+b+1)} \times (M - m)^2$
Mode	$\begin{cases} \frac{a-1}{a+b-2} \times M + \frac{b-1}{a+b-2} \times m & a > 1, b > 1 \\ m \text{ and } M & a < 1, b < 1 \\ m & \begin{cases} a < 1, b \geq 1 \\ a = 1, b > 1 \end{cases} \\ M & \begin{cases} a \geq 1, b < 1 \\ a > 1, b = 1 \end{cases} \\ \text{not unique} & a = b = 1 \end{cases}$
Defaults	SHAPE1=SHAPE2=1, MIN $\to -\infty$, MAX $\to \infty$

Table 4.5 Gamma Distribution

PRIOR statement	GAMMA(SHAPE=a, SCALE=b)
Density	$\frac{1}{b^a \Gamma(a)} \theta^{a-1} e^{-\theta/b}$
Parameter restriction	$a > 0, b > 0$
Range	$[0, \infty)$
Mean	ab
Variance	ab^2
Mode	$(a - 1)b$
Defaults	SHAPE=SCALE=1

Table 4.6 Inverse Gamma Distribution

PRIOR statement	IGAMMA(SHAPE=a, SCALE=b)
Density	$\frac{b^a}{\Gamma(a)}\theta^{-(a+1)}e^{-b/\theta}$
Parameter restriction	$a > 0, b > 0$
Range	$0 < \theta < \infty$
Mean	$\frac{b}{a-1}, \qquad a > 1$
Variance	$\frac{b^2}{(a-1)^2(a-2)}, \qquad a > 2$
Mode	$\frac{b}{a+1}$
Defaults	SHAPE=2.000001, SCALE=1

Table 4.7 Normal Distribution

PRIOR statement	NORMAL(MEAN=μ, VAR=σ^2)
Density	$\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(\theta-\mu)^2}{2\sigma^2}\right)$
Parameter restriction	$\sigma^2 > 0$
Range	$-\infty < \theta < \infty$
Mean	μ
Variance	σ^2
Mode	μ
Defaults	MEAN=0, VAR=1000000

Table 4.8 *t* Distribution

PRIOR statement	T(LOCATION=μ, DF=ν)
Density	$\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma(\frac{\nu}{2})\sqrt{\pi\nu}}\left[1+\frac{(\theta-\mu)^2}{\nu}\right]^{-\frac{\nu+1}{2}}$
Parameter restriction	$\nu > 0$
Range	$-\infty < \theta < \infty$
Mean	μ, for $\nu > 1$
Variance	$\frac{\nu}{\nu-2}$, for $\nu > 2$
Mode	μ
Defaults	LOCATION=0, DF=3

Table 4.9 Uniform Distribution

PRIOR statement	UNIFORM(MIN=m, MAX=M)
Density	$\frac{1}{M-m}$

Parameter restriction	$-\infty < m < M < \infty$
Range	$\theta \in [m, M]$
Mean	$\frac{m+M}{2}$
Variance	$\frac{(M-m)^2}{12}$
Mode	Not unique
Defaults	MIN$\rightarrow -\infty$, MAX$\rightarrow \infty$

Output to SAS Data Set

XBeta, Predicted, and Residual

Xbeta is the structural part on the right-hand side of the model. The predicted value is the predicted dependent variable value. For censored variables, if the predicted value is outside the boundaries, it is reported as the closest boundary. The residual is defined only for continuous variables and is defined as

$$\text{Residual} = \text{Observed} - \text{Predicted}$$

Error Standard Deviation

The error standard deviation is σ_i in the model. It varies only when the HETERO statement is used.

Marginal Effects

A marginal effect is defined as a contribution of one control variable to the response variable. For a binary choice model with two response categories, $\mu_0 = -\infty$ and $\mu_1 = 0$, $\mu_2 = \infty$. For an ordinal response model with M response categories $(\mu_0, \cdots, \mu_M)$, define

$$R_{i,j} = \mu_j - \mathbf{x}_i'\boldsymbol{\beta}$$

The probability that the unobserved dependent variable is contained in the jth category can be written as

$$P[\mu_{j-1} < y_i^* \le \mu_j] = F(R_{i,j}) - F(R_{i,j-1})$$

The marginal effect of changes in the regressors on the probability of $y_i = j$ is then

$$\frac{\partial \text{Prob}[y_i = j]}{\partial \mathbf{x}} = [f(\mu_{j-1} - \mathbf{x}_i'\boldsymbol{\beta}) - f(\mu_j - \mathbf{x}_i'\boldsymbol{\beta})]\boldsymbol{\beta}$$

where $f(x) = \frac{dF(x)}{dx}$. In particular,

$$f(x) = \frac{dF(x)}{dx} = \begin{cases} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} & \text{(probit)} \\ \frac{e^{-x}}{[1+e^{(-x)}]^2} & \text{(logit)} \end{cases}$$

The marginal effects in the truncated regression model are

$$\frac{\partial E[y_i | L_i < y_i^* < R_i]}{\partial \mathbf{x}} = \boldsymbol{\beta}\left[1 - \frac{(\phi(a_i) - \phi(b_i))^2}{(\Phi(b_i) - \Phi(a_i))^2} + \frac{a_i\phi(a_i) - b_i\phi(b_i)}{\Phi(b_i) - \Phi(a_i)}\right]$$

where $a_i = \frac{L_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma_i}$ and $b_i = \frac{R_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma_i}$.

The marginal effects in the censored regression model are

$$\frac{\partial E[y|\mathbf{x}_i]}{\partial \mathbf{x}} = \boldsymbol{\beta} \times \text{Prob}[L_i < y_i^* < R_i]$$

Expected and Conditionally Expected Values

The expected value is the unconditional expectation of the dependent variable. For a censored variable, it is

$$E[y_i] = \Phi(a_i)L_i + (\mathbf{x}_i'\boldsymbol{\beta} + \lambda\sigma_i)(\Phi(b_i) - \Phi(a_i)) + (1 - \Phi(b_i))R_i$$

For a left-censored variable ($R_i = \infty$), this formula is

$$E[y_i] = \Phi(a_i)L_i + (\mathbf{x}_i'\boldsymbol{\beta} + \lambda\sigma_i)(1 - \Phi(a_i))$$

where $\lambda = \frac{\phi(a_i)}{1-\Phi(a_i)}$.

For a right-censored variable ($L_i = -\infty$), this formula is

$$E[y_i] = (\mathbf{x}_i'\boldsymbol{\beta} + \lambda\sigma_i)\Phi(b_i) + (1 - \Phi(b_i))R_i$$

where $\lambda = -\frac{\phi(b_i)}{\Phi(b_i)}$.

For a noncensored variable, this formula is

$$E[y_i] = \mathbf{x}_i'\boldsymbol{\beta}$$

The conditional expected value is the expectation when the variable is inside the boundaries:

$$E[y_i|L_i < y_i < R_i] = \mathbf{x}_i'\boldsymbol{\beta} + \lambda\sigma_i$$

Technical Efficiency

Technical efficiency for each producer is computed only for stochastic frontier models.

In general, the stochastic production frontier can be written as

$$y_i = f(x_i; \beta)\exp\{v_i\}TE_i$$

where y_i denotes producer i's actual output, $f(\cdot)$ is the deterministic part of the production frontier, $\exp\{v_i\}$ is a producer-specific error term, and TE_i is the technical efficiency coefficient, which can be written as

$$TE_i = \frac{y_i}{f(x_i; \beta)\exp\{v_i\}}$$

For a Cobb-Douglas production function, $TE_i = \exp\{-u_i\}$. For more information, see the section "Stochastic Frontier Production and Cost Models" on page 99.

The cost frontier can be written in general as

$$E_i = c(y_i, w_i; \beta)\exp\{v_i\}/CE_i$$

where w_i denotes producer i's input prices, $c(\cdot)$ is the deterministic part of the cost frontier, $\exp\{v_i\}$ is a producer-specific error term, and CE_i is the cost efficiency coefficient, which can be written as

$$CE_i = \frac{c(x_i, w_i; \beta)\exp\{v_i\}}{E_i}$$

For a Cobb-Douglas cost function, $CE_i = \exp\{-u_i\}$. For more information, see the section "Stochastic Frontier Production and Cost Models" on page 99. Hence, both technical and cost efficiency coefficients are the same. The estimates of technical efficiency are provided in the following subsections.

Normal-Half-Normal Model

Define $\mu_* = -\epsilon\sigma_u^2/\sigma^2$ and $\sigma_*^2 = \sigma_u^2\sigma_v^2/\sigma^2$. Then, as shown by Jondrow et al. (1982), conditional density is as follows:

$$f(u|\epsilon) = \frac{f(u,\epsilon)}{f(\epsilon)} = \frac{1}{\sqrt{2\pi}\sigma_*}\exp\left\{-\frac{(u-\mu_*)^2}{2\sigma_*^2}\right\} \bigg/ \left[1-\Phi\left(-\frac{\mu_*}{\sigma_*}\right)\right]$$

Hence, $f(u|\epsilon)$ is the density for $N^+(\mu_*, \sigma_*^2)$.

From this result, it follows that the estimate of technical efficiency (Battese and Coelli 1988) is

$$TE1_i = E(\exp\{-u_i\}|\epsilon_i) = \left[\frac{1-\Phi(\sigma_* - \mu_{*i}/\sigma_*)}{1-\Phi(-\mu_{*i}/\sigma_*)}\right]\exp\left\{-\mu_{*i} + \frac{1}{2}\sigma_*^2\right\}$$

The second version of the estimate (Jondrow et al. 1982) is

$$TE2_i = \exp\{-E(u_i|\epsilon_i)\}$$

where

$$E(u_i|\epsilon_i) = \mu_{*i} + \sigma_*\left[\frac{\phi(-\mu_{*i}/\sigma_*)}{1-\Phi(-\mu_{*i}/\sigma_*)}\right] = \sigma_*\left[\frac{\phi(\epsilon_i\lambda/\sigma)}{1-\Phi(\epsilon_i\lambda/\sigma)} - \left(\frac{\epsilon_i\lambda}{\sigma}\right)\right]$$

Normal-Exponential Model

Define $A = -\tilde{\mu}/\sigma_v$ and $\tilde{\mu} = -\epsilon - \sigma_v^2/\sigma_u$. Then, as shown by Kumbhakar and Lovell (2000), conditional density is as follows:

$$f(u|\epsilon) = \frac{1}{\sqrt{2\pi}\sigma_v\Phi(-\tilde{\mu}/\sigma_v)}\exp\left\{-\frac{(u-\tilde{\mu})^2}{2\sigma^2}\right\}$$

Hence, $f(u|\epsilon)$ is the density for $N^+(\tilde{\mu}, \sigma_v^2)$.

From this result, it follows that the estimate of technical efficiency is

$$TE1_i = E(\exp\{-u_i\}|\epsilon_i) = \left[\frac{1-\Phi(\sigma_v - \tilde{\mu}_i/\sigma_v)}{1-\Phi(-\tilde{\mu}_i/\sigma_v)}\right]\exp\left\{-\tilde{\mu}_i + \frac{1}{2}\sigma_v^2\right\}$$

The second version of the estimate is

$$TE2_i = \exp\{-E(u_i|\epsilon_i)\}$$

where

$$E(u_i|\epsilon_i) = \tilde{\mu}_i + \sigma_v\left[\frac{\phi(-\tilde{\mu}_i/\sigma_v)}{1-\Phi(-\tilde{\mu}_i/\sigma_v)}\right] = \sigma_v\left[\frac{\phi(A)}{\Phi(-A)} - A\right]$$

Normal–Truncated Normal Model

Define $\tilde{\mu} = (-\sigma_u^2 \epsilon_i + \mu\sigma_v^2)/\sigma^2$ and $\sigma_*^2 = \sigma_u^2\sigma_v^2/\sigma^2$. Then, as shown by Kumbhakar and Lovell (2000), conditional density is as follows:

$$f(u|\epsilon) = \frac{1}{\sqrt{2\pi}\sigma_*[1-\Phi(-\tilde{\mu}/\sigma_*)]}\exp\left\{-\frac{(u-\tilde{\mu})^2}{2\sigma_*^2}\right\}$$

Hence, $f(u|\epsilon)$ is the density for $N^+(\tilde{\mu}, \sigma_*^2)$.

From this result, it follows that the estimate of technical efficiency is

$$TE1_i = E(\exp\{-u_i\}|\epsilon_i) = \frac{1-\Phi(\sigma_* - \tilde{\mu}_i/\sigma_*)}{1-\Phi(-\tilde{\mu}_i/\sigma_*)}\exp\left\{-\tilde{\mu}_i + \frac{1}{2}\sigma_*^2\right\}$$

The second version of the estimate is

$$TE2_i = \exp\{-E(u_i|\epsilon_i)\}$$

where

$$E(u_i|\epsilon_i) = \tilde{\mu}_i + \sigma_*\left[\frac{\phi(\tilde{\mu}_i/\sigma_*)}{1-\Phi(-\tilde{\mu}_i/\sigma_*)}\right]$$

OUTEST= Data Set

The OUTEST= data set contains all the parameters that are estimated by a MODEL statement. Each parameter contains the estimate for the corresponding parameter in the corresponding model. In addition, the OUTEST= data set contains the following variables:

NAME	indicates the name of the independent variable.
TYPE	indicates the type of observation. PARM indicates the row of coefficients; STD indicates the row of standard deviations of the corresponding coefficients.
STATUS	indicates the convergence status for optimization.

The rest of the columns correspond to the explanatory variables.

The OUTEST= data set contains one observation for the MODEL statement, which shows the parameter estimates for that model. If you specify the COVOUT option in the PROC HPQLIM statement, the OUTEST= data set includes additional observations for the MODEL statement, which show the rows of the covariance matrix of parameter estimates. For covariance observations, the value of the _TYPE_ variable is COV, and the _NAME_ variable identifies the parameter that is associated with that row of the covariance matrix. If you specify the CORROUT option in the PROC HPQLIM statement, the OUTEST= data set includes additional observations for the MODEL statement, which show the rows of the correlation matrix of parameter estimates. For correlation observations, the value of the _TYPE_ variable is CORR, and the _NAME_ variable identifies the parameter that is associated with that row of the correlation matrix.

Naming

Naming of Parameters

The parameters are named in the same way as in other SAS procedures such as the REG and PROBIT procedures. The constant in the regression equation is called Intercept. The coefficients of independent variables are named by the independent variables. The standard deviation of the errors is called _Sigma. If the HETERO statement is included, the coefficients of the independent variables in the HETERO statement are called _H.x, where x is the name of the independent variable.

Naming of Output Variables

Table 4.10 shows the *options* in the OUTPUT statement, with the corresponding variable names and their explanations.

Table 4.10 OUTPUT Statement Options

output-option	**Variable Name**	**Explanation**
CONDITIONAL	CEXPCT_y	Conditional expected value of y, conditioned on the truncation
ERRSTD	ERRSTD_y	Standard deviation of error term
EXPECTED	EXPCT_y	Unconditional expected value of y
MARGINAL	MEFF_x	Marginal effect of x on y ($\frac{\partial y}{\partial x}$) with single equation
OUTVAR	The variable name used in the input data set	Input data variables are added to the output data set
PREDICTED	P_y	Predicted value of y
RESIDUAL	RESID_y	Residual of y, (y – PredictedY)
TE1	TE1	Technical efficiency estimate for each producer proposed by Battese and Coelli (1988)
TE2	TE2	Technical efficiency estimate for each producer proposed by Jondrow et al. (1982)
XBETA	XBETA_y	Structure part ($\mathbf{x}'\boldsymbol{\beta}$) of y equation

If you prefer to name the output variables differently, you can use the RENAME option in the data set. For example, the following statements rename the residual of y as *Resid*:

```
proc hpqlim data=one;
   model y = x1-x10 / censored;
   output out=outds(rename=(resid_y=resid)) residual;
run;
```

ODS Table Names

PROC HPQLIM assigns a name to each table that it creates. You can use these names to refer to the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 4.11.

Table 4.11 ODS Tables Produced in PROC HPQLIM

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement and TEST Statement		
FitSummary	Summary of nonlinear estimation	Default
ParameterEstimates	Parameter estimates	Default
SummaryContResponse	Summary of continuous response	Default
CovB	Covariance of parameter estimates	COVB
CorrB	Correlation of parameter estimates	CORRB
ODS Tables Created by the BAYES Statement		
AutoCorr	Autocorrelation statistics for each parameter	Default
Corr	Correlation matrix of the posterior samples	STATS=COR
Cov	Covariance matrix of the posterior samples	STATS=COV
ESS	Effective sample size for each parameter	Default
MCSE	Monte Carlo standard error for each parameter	Default
Geweke	Geweke diagnostics for each parameter	Default
Heidelberger	Heidelberger-Welch diagnostics for each parameter	DIAGNOSTICS=HEIDEL
PostIntervals	Equal-tail and HPD intervals for each parameter	Default
PosteriorSample	Posterior samples	(ODS output data set only)
PostSummaries	Posterior summaries	Default
PriorSummaries	Prior summaries	STATS=PRIOR
Raftery	Raftery-Lewis diagnostics for each parameter	DIAGNOSTICS=RAFTERY
ODS Tables Created by the TEST Statement		
TestResults	Test results	Default

ODS Graphics

You can use a name to reference every graph that is produced through ODS Graphics. The names of the graphs that PROC HPQLIM generates are listed in Table 4.12.

Table 4.12 Graphs Produced by PROC HPQLIM When a BAYES Statement Is Included

ODS Graph Name	Plot Description	Statement and Option
Bayesian Diagnostic Plots		
ADPanel	Autocorrelation function and density panel	PLOTS=(AUTOCORR DENSITY)
AutocorrPanel	Autocorrelation function panel	PLOTS=AUTOCORR
AutocorrPlot	Autocorrelation function plot	PLOTS(UNPACK)=AUTOCORR
DensityPanel	Density panel	PLOTS=DENSITY
DensityPlot	Density plot	PLOTS(UNPACK)=DENSITY
TAPanel	Trace and autocorrelation function panel	PLOTS=(TRACE AUTOCORR)
TADPanel	Trace, density, and autocorrelation function panel	PLOTS=(TRACE AUTOCORR DENSITY)
TDPanel	Trace and density panel	PLOTS=(TRACE DENSITY)
TracePanel	Trace panel	PLOTS=TRACE
TracePlot	Trace plot	PLOTS(UNPACK)=TRACE

Examples: The HPQLIM Procedure

Example 4.1: High-Performance Model with Censoring

This example shows the use of the HPQLIM procedure with an emphasis on processing a large data set and on the performance improvements that are achieved by executing in the high-performance distributed environment.

The following DATA step generates 5 million replicates from a censored model. The model contains seven variables.

```
data simulate;
     call streaminit(12345);
     array vars x1-x7;
     array parms{7}  (3 4 2 4 -3 -5 -3);

     intercept=2;

     do i=1 to 5000000;
        sum_xb=0;
        do j=1 to 7;
           vars[j]=rand('NORMAL',0,1);
           sum_xb=sum_xb+parms[j]*vars[j];
        end;
        y=intercept+sum_xb+400*rand('NORMAL',0,1);
        if y>400 then y=400;
        if y<0 then y=0;
        output;
```

```
      end;
   keep y x1-x7;
   run;
```

The following statements estimate a censored model. The model is executed in the distributed computing environment with two threads and only one node. These settings are used to obtain a hypothetical environment that might resemble running the HPQLIM procedure on a desktop workstation with a dual-core CPU. To run these statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to the macro variables in the example with the appropriate values.

```
option set=GRIDHOST="&GRIDHOST";
option set=GRIDINSTALLLOC="&GRIDINSTALLLOC";


proc hpqlim data=simulate ;
   performance nthreads=2 nodes=1 details ;
   model y=x1-x7 /censored(lb=0 ub=400);
run;
```

Output 4.1.1 shows the results for the censored model. The "Performance Information" table shows that the model was estimated on the grid defined in a macro variable named GRIDHOST in a distributed environment on only one node with two threads. The "Model Fit Summary" table shows detailed information about the model and indicates that all 5 million observations were used to fit the model. All parameter estimates in the "Parameter Estimates" table are highly significant and correspond to their theoretical values that were set during the data generating process. The optimization of the model with 5 million observations took 124.47 seconds.

Output 4.1.1 Censored Model with One Node and Two Threads

```
                         Estimating a Tobit model

                           The HPQLIM Procedure

                         Performance Information

            Host Node                      << your grid host >>
            Execution Mode                 Distributed
            Grid Mode                      Symmetric
            Number of Compute Nodes        1
            Number of Threads per Node     2


                            Model Information

                  Data Source                WORK.SIMULATE
                  Response Variable          y
                  Optimization Technique     Quasi-Newton


                         Number of Observations

                Number of Observations Read         5000000
                Number of Observations Used         5000000
```

Output 4.1.1 *continued*

```
                  Summary Statistics of Continuous Responses

                                                                  N Obs N Obs
                      Standard                       Lower     Upper Lower Upper
Variable      Mean       Error        Type           Bound     Bound Bound Bound

   y        127.0   159.491090     Censored             0     400.0 249E4   8E5


             Convergence criterion (FCONV=2.220446E-16) satisfied.


                              Model Fit Summary

                 Number of Endogenous Variables                1
                 Endogenous Variable                           y
                 Number of Observations                  5000000
                 Log Likelihood                        -15268972
                 Maximum Absolute Gradient             0.0003291
                 Number of Iterations                         11
                 Optimization Method                Quasi-Newton
                 AIC                                    30537962
                 Schwarz Criterion                      30538083


                             Parameter Estimates

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

    Intercept      1         2.220379       0.222201       9.99        <.0001
    x1             1         3.055533       0.201620      15.15        <.0001
    x2             1         4.000176       0.201570      19.85        <.0001
    x3             1         1.852740       0.201555       9.19        <.0001
    x4             1         4.170266       0.201533      20.69        <.0001
    x5             1        -3.010679       0.201458     -14.94        <.0001
    x6             1        -5.176016       0.201541     -25.68        <.0001
    x7             1        -2.695948       0.201671     -13.37        <.0001
    _Sigma         1       399.997845       0.261930    1527.12        <.0001


                            Procedure Task Timing

      Task                                        Seconds      Percent

      Reading and Levelizing Data                    3.86        3.01%
      Communication to Client                        0.06        0.05%
      Optimization                                 124.47       96.95%
      Post-optimization                              0.00        0.00%
```

In the following statements, the PERFORMANCE statement is modified to use a grid with 10 nodes, with each node capable of spawning eight threads:

```
 proc hpqlim data=simulate ;
    performance nthreads=8 nodes=10 details ;
    model y=x1-x7 /censored(lb=0 ub=400);
run;
```

Because the two models being estimated are identical, it is reasonable to expect that Output 4.1.1 and Output 4.1.2 would show the same results except for the performance. The second model, which was run on a grid that used 10 nodes with eight threads each, took only 3.53 seconds instead of 124.47 seconds to optimize.

In certain circumstances, you might observe slight numerical differences in the results (depending on the number of nodes and threads) because the order in which partial results are accumulated, the limits of numerical precision, and the propagation of error in numerical computations can make a difference in the final result.

Output 4.1.2 Censored Model on Ten Nodes with Eight Threads Each

```
                           Estimating a Tobit model

                             The HPQLIM Procedure

                           Performance Information

              Host Node                      << your grid host >>
              Execution Mode                 Distributed
              Grid Mode                      Symmetric
              Number of Compute Nodes        10
              Number of Threads per Node     8


                              Model Information

                    Data Source                WORK.SIMULATE
                    Response Variable          y
                    Optimization Technique     Quasi-Newton


                           Number of Observations

                  Number of Observations Read         5000000
                  Number of Observations Used         5000000


                  Summary Statistics of Continuous Responses

                                                                  N Obs N Obs
                       Standard                     Lower    Upper Lower Upper
Variable      Mean        Error        Type          Bound    Bound Bound Bound

    y        127.0   159.491090     Censored            0     400.0 249E4   8E5


             Convergence criterion (FCONV=2.220446E-16) satisfied.
```

Output 4.1.2 *continued*

```
                          Model Fit Summary

              Number of Endogenous Variables                1
              Endogenous Variable                           y
              Number of Observations                  5000000
              Log Likelihood                        -15268972
              Maximum Absolute Gradient             0.0008332
              Number of Iterations                         10
              Optimization Method                Quasi-Newton
              AIC                                    30537962
              Schwarz Criterion                      30538083


                         Parameter Estimates

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

  Intercept     1        2.220358       0.222201       9.99      <.0001
  x1            1        3.055491       0.201620      15.15      <.0001
  x2            1        4.000196       0.201570      19.85      <.0001
  x3            1        1.852735       0.201555       9.19      <.0001
  x4            1        4.170323       0.201533      20.69      <.0001
  x5            1       -3.010670       0.201458     -14.94      <.0001
  x6            1       -5.176019       0.201541     -25.68      <.0001
  x7            1       -2.695886       0.201671     -13.37      <.0001
  _Sigma        1      399.997846       0.261930    1527.12      <.0001


                        Procedure Task Timing

    Task                                          Seconds     Percent

    Reading and Levelizing Data                      0.31       7.78%
    Communication to Client                          0.12       3.10%
    Optimization                                     3.53      89.12%
    Post-optimization                                0.00       0.00%
```

As this example suggests, increasing the number of nodes and the number of threads per node improves performance significantly. When you use the parallelism that a high-performance distributed environment affords, you can see an even more dramatic reduction in the time required for the optimization as the number of observations in the data set increases. When the data set is extremely large, the computations might not even be possible with the typical memory resources and computational constraints of a desktop computer. Under such circumstances the high-performance distributed environment becomes a necessity.

Example 4.2: Bayesian High-Performance Model with Censoring

This example shows the use of the Bayesian analysis available in the HPQLIM procedure with an emphasis on processing a large data set and on the performance improvements that are achieved by executing in a high-performance distributed environment.

The model and the data set are the same as in Example 4.1, and the priors are set to the defaults.

The model is executed in the distributed computing environment with two threads and only one node. These settings are used to obtain a hypothetical environment that might resemble running the HPQLIM procedure on a desktop workstation with a dual-core CPU. To run the following statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to the macro variables in the example with the appropriate values.

```
option set=GRIDHOST="&GRIDHOST";
option set=GRIDINSTALLLOC="&GRIDINSTALLLOC";


proc hpqlim data=simulate ;
bayes nbi=10000 nmc=30000;
   performance nthreads=2 nodes=1 details ;
   model y=x1-x7 /censored(lb=0 ub=400);
run;
```

Output 4.2.1 shows a summary of the posterior distribution that is associated with the censored model when you use diffuse prior distributions.

Output 4.2.1 Posterior Summary for Bayesian Censored Model

```
                              Estimating a Tobit model

                                The HPQLIM Procedure

                                Posterior Summaries

                                       Standard               Percentiles
Parameter          N         Mean     Deviation         25%          50%          75%

Intercept      30000       2.2104        0.2219      2.0605       2.2115       2.3629
x1             30000       3.0565        0.1991      2.9205       3.0585       3.1925
x2             30000       4.0045        0.1992      3.8706       4.0054       4.1342
x3             30000       1.8466        0.1993      1.7142       1.8502       1.9823
x4             30000       4.1695        0.1965      4.0355       4.1700       4.3059
x5             30000      -3.0167        0.1983     -3.1511      -3.0167      -2.8843
x6             30000      -5.1583        0.1895     -5.2845      -5.1611      -5.0255
x7             30000      -2.6868        0.2038     -2.8267      -2.6860      -2.5529
_Sigma         30000        400.0        0.2597       399.8        400.0        400.2
```

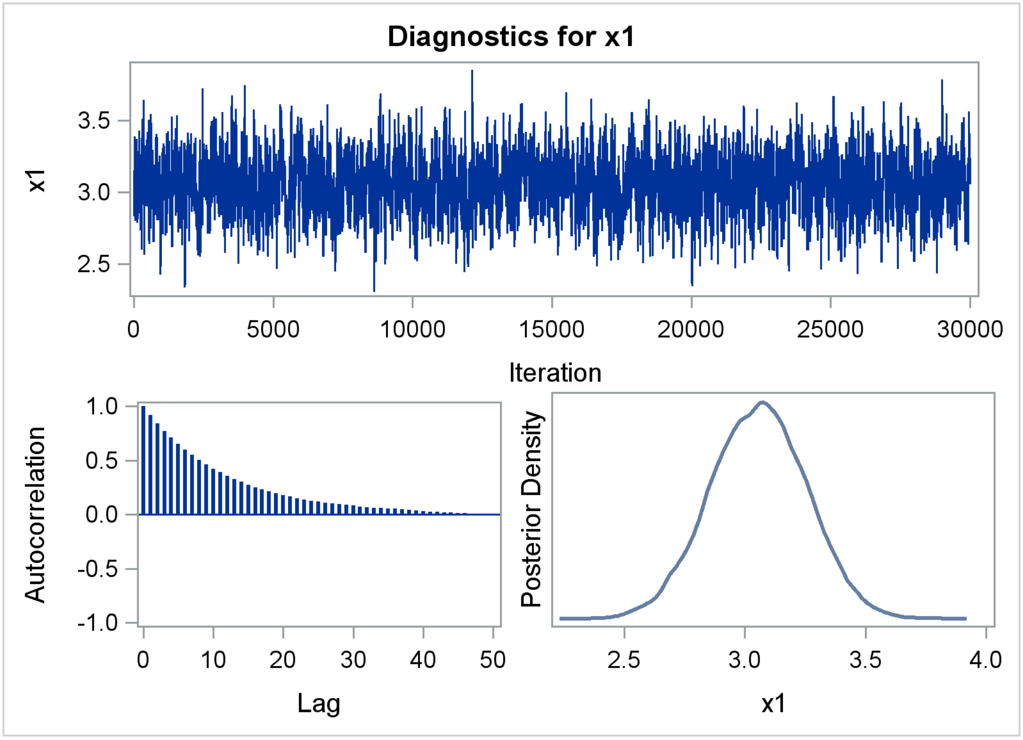
Output 4.2.2 show a summary of the performance when you use a distributed computing environment with one node and two threads.

Output 4.2.2 Performance Analysis for Bayesian Censored Model on One Node with Two Threads

```
                         Estimating a Tobit model

                           The HPQLIM Procedure

                         Performance Information

              Host Node                     << your grid host >>
              Execution Mode                Distributed
              Grid Mode                     Symmetric
              Number of Compute Nodes       1
              Number of Threads per Node    2


                         Estimating a Tobit model

                           The HPQLIM Procedure

                          Procedure Task Timing

       Task                                          Seconds     Percent

       Reading and Levelizing Data                      3.73       0.00%
       Communication to Client                          0.15       0.00%
       Bayesian Analysis: Likelihood for MCMC       86759.60      99.85%
       Bayesian Analysis: MCMC                          1.15       0.00%
       Optimization                                   124.95       0.14%
       Post-optimization                                0.00       0.00%
```

Finally, Output 4.2.3 shows the diagnostic and summary plots that are associated with *X1*.

Output 4.2.3 Bayesian Diagnostic and Summary Plots for x1



In the following statements, the PERFORMANCE statement is modified to use a grid with 10 nodes, where each node spawns eight threads:

```
option set=GRIDHOST="&GRIDHOST";
option set=GRIDINSTALLLOC="&GRIDINSTALLLOC";


proc hpqlim data=simulate ;
bayes nbi=10000 nmc=30000;
   performance nthreads=8 nodes=10 details ;
   model y=x1-x7 /censored(lb=0 ub=400);
run;
```

The two models are identical, but the second implementation, which was run on a grid that used 10 nodes with eight threads each, took only 49 minutes instead of 24 hours and 6 minutes to sample from the same posterior distribution.

Output 4.2.4 Performance Analysis for Bayesian Censored Model on Ten Nodes with Eight Threads Each

```
                           Estimating a Tobit model

                             The HPQLIM Procedure

                           Performance Information

              Host Node                     << your grid host >>
              Execution Mode                Distributed
              Grid Mode                     Symmetric
              Number of Compute Nodes       10
              Number of Threads per Node    8


                           Estimating a Tobit model

                             The HPQLIM Procedure

                            Procedure Task Timing

       Task                                          Seconds     Percent

       Reading and Levelizing Data                      0.31       0.01%
       Communication to Client                          0.17       0.01%
       Bayesian Analysis: Likelihood for MCMC        2921.57      99.85%
       Bayesian Analysis: MCMC                          0.55       0.02%
       Optimization                                     3.43       0.12%
       Post-optimization                                0.00       0.00%
```

References

Aigner, C., Lovell, C. A. K., and Schmidt, P. (1977), "Formulation and Estimation of Stochastic Frontier Production Function Models," *Journal of Econometrics*, 6, 21–37.

Battese, G. E. and Coelli, T. J. (1988), "Prediction of Firm-Level Technical Efficiencies with a Generalized Frontier Production Function and Panel Data," *Journal of Econometrics*, 38, 387–399.

Christensen, L. R. and Greene, W. H. (1976), "Economies of Scale in U.S. Electric Power Generation," *Journal of Political Economy*, 84, 655–676.

Coelli, T. J., Prasada Rao, D. S., and Battese, G. E. (1998), *An Introduction to Efficiency and Productivity Analysis*, London: Kluwer Academic.

Gallant, A. R. (1987), *Nonlinear Statistical Models*, New York: John Wiley & Sons.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2004), *Bayesian Data Analysis*, 2nd Edition, London: Chapman & Hall.

Gregory, A. W. and Veall, M. R. (1985), "On Formulating Wald Tests for Nonlinear Restrictions," *Econometrica*, 53, 1465–1468.

Jondrow, J., Lovell, C. A. K., Materov, I. S., and Schmidt, P. (1982), "On the Estimation of Technical Efficiency in the Stochastic Frontier Production Function Model," *Journal of Econometrics*, 19, 233–238.

Kumbhakar, S. C. and Lovell, C. A. K. (2000), *Stochastic Frontier Analysis*, New York: Cambridge University Press.

Meeusen, W. and van den Broeck, J. (1977), "Efficiency Estimation from Cobb-Douglas Production Functions with Composed Error," *International Economic Review*, 18, 435–444.

Mroz, T. A. (1987), "The Sensitivity of an Empirical Model of Married Women's Work to Economic and Statistical Assumptions," *Econometrica*, 55, 765–799.

Phillips, C. B. and Park, J. Y. (1988), "On Formulating Wald Tests of Nonlinear Restrictions," *Econometrica*, 56, 1065–1083.

Roberts, G. O., Gelman, A., and Gilks, W. R. (1997), "Weak Convergence and Optimal Scaling of Random Walk Metropolis Algorithms," *Annals of Applied Probability*, 7, 110–120.

Roberts, G. O. and Rosenthal, J. S. (2001), "Optimal Scaling for Various Metropolis-Hastings Algorithms," *Statistical Science*, 16, 351–367.

Schervish, M. J. (1995), *Theory of Statistics*, New York: Springer-Verlag.

Wooldridge, J. M. (2002), *Econometric Analysis of Cross Section of Panel Data*, Cambridge, MA: MIT Press.

Chapter 5
The HPSEVERITY Procedure

Contents

Overview: HPSEVERITY Procedure

PROC HPSEVERITY estimates parameters of any arbitrary continuous probability distribution that is used to model the magnitude (severity) of a continuous-valued event of interest. Some examples of such events are loss amounts paid by an insurance company and demand of a product as depicted by its sales. PROC HPSEVERITY is especially useful when the severity of an event does not follow typical distributions (such as the normal distribution) that are often assumed by standard statistical methods.

PROC HPSEVERITY runs in either single-machine mode or distributed mode. **NOTE:** Distributed mode requires SAS High-Performance Econometrics.

PROC HPSEVERITY provides a default set of probability distribution models that includes the Burr, exponential, gamma, generalized Pareto, inverse Gaussian (Wald), lognormal, Pareto, Tweedie, and Weibull distributions. In the simplest form, you can estimate the parameters of any of these distributions by using a list of severity values that are recorded in a SAS data set. PROC HPSEVERITY computes the estimates of the model parameters, their standard errors, and their covariance structure by using the maximum likelihood method.

PROC HPSEVERITY can fit multiple distributions at the same time and choose the best distribution according to a specified selection criterion. Seven different statistics of fit can be used as selection criteria. They are log likelihood, Akaike's information criterion (AIC), corrected Akaike's information criterion (AICC), Schwarz Bayesian information criterion (BIC), Kolmogorov-Smirnov statistic (KS), Anderson-Darling statistic (AD), and Cramér-von Mises statistic (CvM).

You can request the procedure to output the status of the estimation process, the parameter estimates and their standard errors, the estimated covariance structure of the parameters, and the statistics of fit.

The following key features make PROC HPSEVERITY unique among SAS procedures that can estimate continuous probability distributions:

- It enables you to fit a distribution model when the severity values are truncated or censored or both. You can specify any combination of the following types of censoring and truncation effects: left-censoring, right-censoring, left-truncation, or right-truncation. This is especially useful in applications with an insurance-type model where a severity (loss) is reported and recorded only if it is greater than the deductible amount (left-truncation) and where a severity value greater than or equal to the policy limit is recorded at the limit (right-censoring). Another useful application is that of interval-censored data, where you know both the lower limit (right-censoring) and upper limit (left-censoring) on the severity, but you do not know the exact value.

  It uses an appropriate estimator of the empirical distribution function (EDF). EDF is required to compute the KS, AD, and CvM statistics-of-fit. The procedures also provide the EDF estimates to your custom parameter initialization method. When truncation or censoring is specified, the EDF is estimated by using either Kaplan-Meier's product-limit estimator or Turnbull's estimator. The former is used by default when only one form of censoring effect (right-censoring or left-censoring) is specified, whereas the latter is used by default when both left-censoring and right-censoring effects are specified.

- It enables you to define any arbitrary continuous parametric distribution model and to estimate its parameters. You just need to define the key components of the distribution, such as its probability density function (PDF) and cumulative distribution function (CDF), as a set of functions and subroutines written with the FCMP procedure, which is part of Base SAS software. As long as the functions and subroutines follow certain rules, the HPSEVERITY procedure can fit the distribution model defined by them.

- It can model the effect of exogenous or regressor variables on a probability distribution, as long as the distribution has a scale parameter. A linear combination of the regressor variables is assumed to affect the scale parameter via an exponential link function.

  If a distribution does not have a scale parameter, then either it needs to have another parameter that can be derived from a scale parameter by using a supported transformation or it needs to be reparameterized to have a scale parameter. If neither of these is possible, then regression effects cannot be modeled.

- It enables you to specify your own objective function to be optimized for estimating the parameters of a model. You can write SAS programming statements to specify the contribution of each observation to the objective function. You can use keyword functions such as _PDF_ and _CDF_ to generalize the objective function to any distribution. If you do not specify your own objective function, then the parameters of a model are estimated by maximizing the likelihood function of the data.

Because the HPSEVERITY procedure is a high-performance analytical procedure, it also does the following:

- enables you to run in distributed mode on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on the server where SAS is installed
- exploits all the available cores and concurrent threads, regardless of execution mode

For more information, see the section "Processing Modes" on page 6 in Chapter 2, "Shared Concepts and Topics.".

Getting Started: HPSEVERITY Procedure

This section outlines the use of the HPSEVERITY procedure to fit continuous probability distribution models. Three examples illustrate different features of the procedure.

A Simple Example of Fitting Predefined Distributions

The simplest way to use PROC HPSEVERITY is to fit all the predefined distributions to a set of values and let the procedure identify the best fitting distribution.

Consider a lognormal distribution, whose probability density function (PDF) f and cumulative distribution function (CDF) F are as follows, respectively, where Φ denotes the CDF of the standard normal distribution:

$$f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\log(x)-\mu}{\sigma}\right)^2} \quad \text{and} \quad F(x;\mu,\sigma) = \Phi\left(\frac{\log(x)-\mu}{\sigma}\right)$$

The following DATA step statements simulate a sample from a lognormal distribution with population parameters $\mu = 1.5$ and $\sigma = 0.25$, and store the sample in the variable Y of a data set Work.Test_sev1:

```
/*------------- Simple Lognormal Example -------------*/
data test_sev1(keep=y label='Simple Lognormal Sample');
   call streaminit(45678);
   label y='Response Variable';
   Mu = 1.5;
   Sigma = 0.25;
   do n = 1 to 100;
      y = exp(Mu) * rand('LOGNORMAL')**Sigma;
      output;
   end;
run;
```

The following statements fit all the predefined distribution models to the values of Y and identify the best distribution according to the corrected Akaike's information criterion (AICC):

```
proc hpseverity data=test_sev1 crit=aicc;
   loss y;
   dist _predefined_;
run;
```

The PROC HPSEVERITY statement specifies the input data set along with the model selection criterion, the LOSS statement specifies the variable to be modeled, and the DIST statement with the _PREDEFINED_ keyword specifies that all the predefined distribution models be fitted.

Some of the default output displayed by this step is shown in Figure 5.1 through Figure 5.3. First, information about the input data set is displayed followed by the "Model Selection" table, as shown in Figure 5.1. The model selection table displays the convergence status, the value of the selection criterion, and the selection status for each of the candidate models. The Converged column indicates whether the estimation process for a given distribution model has converged, might have converged, or failed. The Selected column indicates whether a given distribution has the best fit for the data according to the selection criterion. For this example, the lognormal distribution model is selected, because it has the lowest value for the selection criterion.

Figure 5.1 Data Set Information and Model Selection Table

```
                    The HPSEVERITY Procedure

                         Input Data Set

                Name                WORK.TEST_SEV1
                Label     Simple Lognormal Sample
```

Figure 5.1 *continued*

```
                      Model Selection

   Distribution    Converged          AICC    Selected

   Burr            Yes           322.50845    No
   Exp             Yes           508.12287    No
   Gamma           Yes           320.50264    No
   Igauss          Yes           319.61652    No
   Logn            Yes           319.56579    Yes
   Pareto          Yes           510.28172    No
   Gpd             Yes           510.20576    No
   Weibull         Yes           334.82373    No
```

Next, the estimation information for each of the candidate models is displayed. The information for the lognormal model, which is the best fitting model, is shown in Figure 5.2. The first table displays a summary of the distribution. The second table displays the convergence status. This is followed by a summary of the optimization process which indicates the technique used, the number of iterations, the number of times the objective function was evaluated, and the log likelihood attained at the end of the optimization. Since the model with lognormal distribution has converged, PROC HPSEVERITY displays its statistics of fit and parameter estimates. The estimates of *Mu*=1.49605 and *Sigma*=0.26243 are quite close to the population parameters of *Mu*=1.5 and *Sigma*=0.25 from which the sample was generated. The p-value for each estimate indicates the rejection of the null hypothesis that the estimate is 0, implying that both the estimates are significantly different from 0.

Figure 5.2 Estimation Details for the Lognormal Model

```
                   The HPSEVERITY Procedure
                      Logn Distribution

                   Distribution Information

    Name                                            Logn
    Description                     Lognormal Distribution
    Distribution Parameters                            2


                      Convergence Status

      Convergence criterion (GCONV=1E-8) satisfied.


                     Optimization Summary

          Optimization Technique    Trust Region
          Iterations                           2
          Function Calls                       8
          Log Likelihood              -157.72104
```

Figure 5.2 *continued*

```
                              Fit Statistics

                   -2 Log Likelihood      315.44208
                   AIC                    319.44208
                   AICC                   319.56579
                   BIC                    324.65242
                   Kolmogorov-Smirnov       0.50641
                   Anderson-Darling         0.31240
                   Cramer-von Mises         0.04353


                           Parameter Estimates

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

            Mu              1.49605       0.02651      56.43      <.0001
            Sigma           0.26243       0.01874      14.00      <.0001
```

The parameter estimates of the Burr distribution are shown in Figure 5.3. These estimates are used in the next example.

Figure 5.3 Parameter Estimates for the Burr Model

```
                           Parameter Estimates

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

            Theta           4.62348       0.46181      10.01      <.0001
            Alpha           1.15706       0.47493       2.44      0.0167
            Gamma           6.41227       0.99039       6.47      <.0001
```

An Example with Left-Truncation and Right-Censoring

PROC HPSEVERITY enables you to specify that the response variable values are left-truncated or right-censored. The following DATA step expands the data set of the previous example to simulate a scenario that is typically encountered by an automobile insurance company. The values of the variable Y represent the loss values on claims that are reported to an auto insurance company. The variable THRESHOLD records the deductible on the insurance policy. If the actual value of Y is less than or equal to the deductible, then it is unobservable and does not get recorded. In other words, THRESHOLD specifies the left-truncation of Y. LIMIT records the policy limit. If the value of Y is equal to or greater than the recorded value, then the observation is right-censored.

```
/*----- Lognormal Model with left-truncation and censoring -----*/
data test_sev2(keep=y threshold limit
        label='A Lognormal Sample With Censoring and Truncation');
   set test_sev1;
   label y='Censored & Truncated Response';
   if _n_ = 1 then call streaminit(45679);

   /* make about 20% of the observations left-truncated */
   if (rand('UNIFORM') < 0.2) then
      threshold = y * (1 - rand('UNIFORM'));
   else
      threshold = .;
   /* make about 15% of the observations right-censored */
   iscens = (rand('UNIFORM') < 0.15);
   if (iscens) then
      limit = y;
   else
      limit = .;
run;
```

The following statements use the AICC criterion to analyze which of the four predefined distributions (lognormal, Burr, gamma, and Weibull) has the best fit for the data:

```
proc hpseverity data=test_sev2 crit=aicc print=all ;
   loss y / lt=threshold rc=limit;

   dist logn burr gamma weibull;
   performance nthreads=2;
run;
```

The LOSS statement specifies the left-truncation and right-censoring variables. Each candidate distribution needs to be specified by using a separate DIST statement. The PRINT= option in the PROC HPSEVERITY statement requests that all the displayed output be prepared. The NTHREADS option in the PERFORMANCE statement specifies that two threads of computation be used. The option is shown here just for illustration. You should use it only when you want to restrict the procedure to use a different number of threads than the value of the CPUCOUNT= system option, which usually defaults to the number of physical CPU cores available on your machine, thereby allowing the procedure to fully utilize the computational power of your machine.

Some of the key results prepared by PROC HPSEVERITY are shown in Figure 5.4 through Figure 5.7. In addition to the estimates of the range, mean, and standard deviation of Y, the “Descriptive Statistics for Variable y” table shown in Figure 5.4 also indicates the number of observations that are left-truncated or right-censored. The “Model Selection” table in Figure 5.4 shows that models with all the candidate distributions have converged and that the Logn (lognormal) model has the best fit for the data according to the AICC criterion.

Figure 5.4 Summary Results for the Truncated and Censored Data

```
                    The HPSEVERITY Procedure

                        Input Data Set

      Name                                      WORK.TEST_SEV2
      Label    A Lognormal Sample With Censoring and Truncation


                  Descriptive Statistics for y

            Observations                              100
            Observations Used for Estimation          100
            Minimum                               2.30264
            Maximum                               8.34116
            Mean                                  4.62007
            Standard Deviation                    1.23627
            Left Truncated Observations                23
            Right Censored Observations                14


                         Model Selection

         Distribution    Converged          AICC    Selected

         Logn            Yes           298.92672    Yes
         Burr            Yes           302.66229    No
         Gamma           Yes           299.45293    No
         Weibull         Yes           309.26779    No
```

PROC HPSEVERITY also prepares a table that shows all the fit statistics for all the candidate models. It is useful to see which model would be the best fit according to each of the criteria. The "All Fit Statistics" table prepared for this example is shown in Figure 5.5. It indicates that the lognormal model is chosen by all the criteria.

Figure 5.5 Comparing All Statistics of Fit for the Truncated and Censored Data

```
                              All Fit Statistics

                      -2 Log
 Distribution       Likelihood          AIC          AICC           BIC            KS

 Logn               294.80301*    298.80301*    298.92672*    304.01335*      0.51824*
 Burr               296.41229     302.41229     302.66229     310.22780       0.66984
 Gamma              295.32921     299.32921     299.45293     304.53955       0.62511
 Weibull            305.14408     309.14408     309.26779     314.35442       0.93307

                    Note: The asterisk (*) marks the best model
                       according to each column's criterion.

                               All Fit Statistics

                   Distribution           AD            CvM

                   Logn               0.34736*       0.05159*
                   Burr               0.36712        0.05726
                   Gamma              0.42921        0.05526
                   Weibull            1.40699        0.17465

                    Note: The asterisk (*) marks the best
                           model according to each
                             column's criterion.
```

Specifying Initial Values for Parameters

All the predefined distributions have parameter initialization functions built into them. For the current example, Figure 5.6 shows the initial values that are obtained by the predefined method for the Burr distribution. It also shows the summary of the optimization process and the final parameter estimates.

Figure 5.6 Burr Model Summary for the Truncated and Censored Data

```
                  Initial Parameter Values and Bounds

                          Initial          Lower         Upper
             Parameter      Value          Bound         Bound

             Theta        4.78102     1.05367E-8         Infty
             Alpha        2.00000     1.05367E-8         Infty
             Gamma        2.00000     1.05367E-8         Infty


                          Optimization Summary

                 Optimization Technique    Trust Region
                 Iterations                           8
                 Function Calls                      23
                 Log Likelihood              -148.20614
```

Figure 5.6 *continued*

```
                         Parameter Estimates

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

     Theta            4.76980     0.62492       7.63      <.0001
     Alpha            1.16363     0.58859       1.98      0.0509
     Gamma            5.94081     1.05004       5.66      <.0001
```

You can specify a different set of initial values if estimates are available from fitting the distribution to similar data. For this example, the parameters of the Burr distribution can be initialized with the final parameter estimates of the Burr distribution that were obtained in the first example (shown in Figure 5.3). One of the ways in which you can specify the initial values is as follows:

```
/*------ Specifying initial values using INIT= option -------*/
proc hpseverity data=test_sev2 crit=aicc print=all;
   loss y / lt=threshold rc=limit;

   dist burr(init=(theta=4.62348 alpha=1.15706 gamma=6.41227));
   performance nthreads=2;
run;
```

The names of the parameters specified in the INIT option must match the names used in the definition of the distribution. The results obtained with these initial values are shown in Figure 5.7. These results indicate that new set of initial values causes the optimizer to reach the same solution with fewer iterations and function evaluations as compared to the default initialization.

Figure 5.7 Burr Model Optimization Summary for the Truncated and Censored Data

```
                       The HPSEVERITY Procedure
                          Burr Distribution

                         Optimization Summary

               Optimization Technique    Trust Region
               Iterations                           5
               Function Calls                      16
               Log Likelihood              -148.20614


                         Parameter Estimates

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

     Theta            4.76980     0.62492       7.63      <.0001
     Alpha            1.16363     0.58859       1.98      0.0509
     Gamma            5.94081     1.05004       5.66      <.0001
```

An Example of Modeling Regression Effects

Consider a scenario in which the magnitude of the response variable might be affected by some regressor (exogenous or independent) variables. The HPSEVERITY procedure enables you to model the effect of such variables on the distribution of the response variable via an exponential link function. In particular, if you have k random regressor variables denoted by x_j $(j = 1, \ldots, k)$, then the distribution of the response variable Y is assumed to have the form

$$Y \sim \exp(\sum_{j=1}^{k} \beta_j x_j) \cdot \mathcal{F}(\Theta)$$

where $\mathcal{F}$ denotes the distribution of Y with parameters Θ and $\beta_j (j = 1, \ldots, k)$ denote the regression parameters (coefficients).

For the effective distribution of Y to be a valid distribution from the same parametric family as $\mathcal{F}$, it is necessary for $\mathcal{F}$ to have a scale parameter. The effective distribution of Y can be written as

$$Y \sim \mathcal{F}(\theta, \Omega)$$

where θ denotes the scale parameter and Ω denotes the set of nonscale parameters. The scale θ is affected by the regressors as

$$\theta = \theta_0 \cdot \exp(\sum_{j=1}^{k} \beta_j x_j)$$

where θ_0 denotes a *base* value of the scale parameter.

Given this form of the model, PROC HPSEVERITY allows a distribution to be a candidate for modeling regression effects only if it has an untransformed or a log-transformed scale parameter.

All the predefined distributions, except the lognormal distribution, have a direct scale parameter (that is, a parameter that is a scale parameter without any transformation). For the lognormal distribution, the parameter μ is a log-transformed scale parameter. This can be verified by replacing μ with a parameter $\theta = e^{\mu}$, which results in the following expressions for the PDF f and the CDF F in terms of θ and σ, respectively, where Φ denotes the CDF of the standard normal distribution:

$$f(x; \theta, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\log(x) - \log(\theta)}{\sigma}\right)^2} \quad \text{and} \quad F(x; \theta, \sigma) = \Phi\left(\frac{\log(x) - \log(\theta)}{\sigma}\right)$$

With this parameterization, the PDF satisfies the $f(x; \theta, \sigma) = \frac{1}{\theta} f(\frac{x}{\theta}; 1, \sigma)$ condition and the CDF satisfies the $F(x; \theta, \sigma) = F(\frac{x}{\theta}; 1, \sigma)$ condition. This makes θ a scale parameter. Hence, $\mu = \log(\theta)$ is a log-transformed scale parameter and the lognormal distribution is eligible for modeling regression effects.

The following DATA step simulates a lognormal sample whose scale is decided by the values of the three regressors X1, X2, and X3 as follows:

$$\mu = \log(\theta) = 1 + 0.75 \text{ X1} - \text{X2} + 0.25 \text{ X3}$$

```
/*----------- Lognormal Model with Regressors ------------*/
data test_sev3(keep=y x1-x3
               label='A Lognormal Sample Affected by Regressors');
   array x{*} x1-x3;
   array b{4} _TEMPORARY_ (1 0.75 -1 0.25);
   call streaminit(45678);
   label y='Response Influenced by Regressors';
   Sigma = 0.25;
   do n = 1 to 100;
      Mu = b(1); /* log of base value of scale */
      do i = 1 to dim(x);
         x(i) = rand('UNIFORM');
         Mu = Mu + b(i+1) * x(i);
      end;
      y = exp(Mu) * rand('LOGNORMAL')**Sigma;
      output;
   end;
run;
```

The following PROC HPSEVERITY step fits the lognormal, Burr, and gamma distribution models to this data. The regressors are specified in the SCALEMODEL statement.

```
proc hpseverity data=test_sev3 crit=aicc print=all;
   loss y;
   scalemodel x1-x3;

   dist logn burr gamma;
run;
```

Some of the key results prepared by PROC HPSEVERITY are shown in Figure 5.8 through Figure 5.12. The descriptive statistics of all the variables are shown in Figure 5.8.

Figure 5.8 Summary Results for the Regression Example

```
                      The HPSEVERITY Procedure

                           Input Data Set

                Name                             WORK.TEST_SEV3
                Label    A Lognormal Sample Affected by Regressors


                     Descriptive Statistics for y

                  Observations                               100
                  Observations Used for Estimation           100
                  Minimum                                1.17863
                  Maximum                                6.65269
                  Mean                                   2.99859
                  Standard Deviation                     1.12845
```

Figure 5.8 *continued*

```
                    Descriptive Statistics for Regressors

                                                                   Standard
Variable              N       Minimum       Maximum          Mean     Deviation

x1                  100     0.0005115       0.97971       0.51689       0.28206
x2                  100       0.01883       0.99937       0.47345       0.28885
x3                  100       0.00255       0.97558       0.48301       0.29709
```

The comparison of the fit statistics of all the models is shown in Figure 5.9. It indicates that the lognormal model is the best model according to each of the likelihood-based statistics.

Figure 5.9 Comparison of Statistics of Fit for the Regression Example

```
                             All Fit Statistics

                   -2 Log
Distribution   Likelihood          AIC         AICC          BIC            KS

Logn           187.49609*   197.49609*   198.13439*   210.52194*      1.97544
Burr           190.69154    202.69154    203.59476    218.32256       2.09334
Gamma          188.91483    198.91483    199.55313    211.94069       1.94472*

                 Note: The asterisk (*) marks the best model
                    according to each column's criterion.

                              All Fit Statistics

                   Distribution          AD            CvM

                   Logn            17.24618        1.21665
                   Burr            13.93436*       1.28529
                   Gamma           15.84787        1.17617*

                    Note: The asterisk (*) marks the best
                          model according to each
                            column's criterion.
```

The distribution information and the convergence results of the lognormal model are shown in Figure 5.10. The iteration history gives you a summary of how the optimizer is traversing the surface of the log-likelihood function in its attempt to reach the optimum. Both the change in the log likelihood and the maximum gradient of the objective function with respect to any of the parameters typically approach 0 if the optimizer converges.

Figure 5.10 Convergence Results for the Lognormal Model with Regressors

```
                         The HPSEVERITY Procedure
                            Logn Distribution

                        Distribution Information

            Name                                          Logn
            Description                 Lognormal Distribution
            Distribution Parameters                          2
            Regression Parameters                            3


                           Convergence Status

              Convergence criterion (GCONV=1E-8) satisfied.


                      Optimization Iteration History

                  Function          -Log                       Maximum
          Iter       Calls    Likelihood          Change      Gradient

             0           2      93.75285                       6.16002
             1           4      93.74805      -0.0048055       0.11031
             2           6      93.74805      -1.5017E-6    0.00003376
             3          10      93.74805      -1.279E-13    3.1051E-12


                          Optimization Summary

                Optimization Technique    Trust Region
                Iterations                           3
                Function Calls                      10
                Log Likelihood               -93.74805
```

The final parameter estimates of the lognormal model are shown in Figure 5.11. All the estimates are significantly different from 0. The estimate that is reported for the parameter *Mu* is the base value for the log-transformed scale parameter μ. Let $x_i (1 \le i \le 3)$ denote the observed value for regressor Xi. If the lognormal distribution is chosen to model Y, then the effective value of the parameter μ varies with the observed values of regressors as

$$\mu = 1.04047 + 0.65221\, x_1 - 0.91116\, x_2 + 0.16243\, x_3$$

These estimated coefficients are reasonably close to the population parameters (that is, within one or two standard errors).

Figure 5.11 Parameter Estimates for the Lognormal Model with Regressors

```
                         Parameter Estimates

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

    Mu               1.04047       0.07614      13.66      <.0001
    Sigma            0.22177       0.01609      13.78      <.0001
    x1               0.65221       0.08167       7.99      <.0001
    x2              -0.91116       0.07946     -11.47      <.0001
    x3               0.16243       0.07782       2.09      0.0395
```

The estimates of the gamma distribution model, which is the next best model according to the fit statistics, are shown in Figure 5.12. The estimate that is reported for the parameter *Theta* is the base value for the scale parameter θ. If the gamma distribution is chosen to model Y, then the effective value of the scale parameter is $\theta = 0.14293\ \exp(0.64562\, x_1 - 0.89831\, x_2 + 0.14901\, x_3)$.

Figure 5.12 Parameter Estimates for the Gamma Model with Regressors

```
                         Parameter Estimates

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

    Theta            0.14293       0.02329       6.14      <.0001
    Alpha           20.37726       2.93277       6.95      <.0001
    x1               0.64562       0.08224       7.85      <.0001
    x2              -0.89831       0.07962     -11.28      <.0001
    x3               0.14901       0.07870       1.89      0.0613
```

Syntax: HPSEVERITY Procedure

The following statements are available in the HPSEVERITY procedure:

PROC HPSEVERITY *options* ;
  LOSS < *response-variable* > < / *censoring-truncation-options* > ;
  WEIGHT *weight-variable* ;
  SCALEMODEL *regressor-variable-list* < / *scalemodel-option* > ;
  DIST *distribution-name-or-keyword* < (*distribution-option*) < *distribution-name-or-keyword* < (*distribution-option*) > > ... > < / *preprocess-options* > ;
  NLOPTIONS *options* ;
  PERFORMANCE *options* ;
  Programming statements ;

Functional Summary

Table 5.1 summarizes the statements and options that control the HPSEVERITY procedure.

Table 5.1 HPSEVERITY Functional Summary

Description	Statement	Option
Statements		
Specifies the response variable to model along with censoring and truncation effects	LOSS	
Specifies the weight variable	WEIGHT	
Specifies the regression variables to model	SCALEMODEL	
Specifies distributions to fit	DIST	
Specifies optimization options	NLOPTIONS	
Specifies performance options	PERFORMANCE	
Specifies programming statements that define an objective function	Programming statements	
Data Set Options		
Specifies that the OUTEST= data set contain covariance estimates	PROC HPSEVERITY	COVOUT
Specifies the input data set	PROC HPSEVERITY	DATA=
Specifies the input data set for parameter estimates	PROC HPSEVERITY	INEST=
Specifies the output data set for parameter estimates	PROC HPSEVERITY	OUTEST=
Specifies the output data set for model information	PROC HPSEVERITY	OUTMODELINFO=
Specifies the output data set for statistics of fit	PROC HPSEVERITY	OUTSTAT=
Data Interpretation Options		
Specifies left-censoring	LOSS	LEFTCENSORED=
Specifies left-truncation	LOSS	LEFTTRUNCATED=
Specifies right-censoring	LOSS	RIGHTCENSORED=
Specifies right-truncation	LOSS	RIGHTTRUNCATED=
Model Estimation Options		
Specifies the model selection criterion	PROC HPSEVERITY	CRITERION=
Specifies the method for computing mixture distribution	SCALEMODEL	DFMIXTURE=
Specifies initial values for model parameters	DIST	INIT=
Specifies the objective function symbol	PROC HPSEVERITY	OBJECTIVE=
Specifies the optimization technique	NLOPTIONS	TECHNIQUE=
Specifies the denominator for computing covariance estimates	PROC HPSEVERITY	VARDEF=
Optimization Termination Criteria		
Absolute function value criterion	NLOPTIONS	ABSCONV=
Absolute function difference convergence criterion	NLOPTIONS	ABSFCONV=

Table 5.1 *continued*

Description	Statement	Option
Absolute gradient convergence criterion	NLOPTIONS	ABSGCONV=
Absolute parameter convergence criterion	NLOPTIONS	ABSXCONV=
Relative function convergence criterion	NLOPTIONS	FCONV=
Another relative function convergence criterion	NLOPTIONS	FCONV2=
Used in FCONV, GCONV criterion	NLOPTIONS	FSIZE=
Relative gradient convergence criterion	NLOPTIONS	GCONV=
Maximum number of function calls	NLOPTIONS	MAXFUNC=
Maximum number of iterations	NLOPTIONS	MAXITER=
Upper limit on CPU time in seconds	NLOPTIONS	MAXTIME=
Minimum number of iterations	NLOPTIONS	MINITER=
Relative parameter convergence criterion	NLOPTIONS	XCONV=
Used in XCONV criterion	NLOPTIONS	XSIZE=
Empirical Distribution Function (EDF) Estimation Options		
Specifies the nonparametric method of CDF estimation	PROC HPSEVERITY	EMPIRICALCDF=
Specifies the sample to be used for computing the EDF estimates	PROC HPSEVERITY	INITSAMPLE
EMPIRICALCDF=MODIFIEDKM Options		
Specifies the α value for the lower bound on risk set size	PROC HPSEVERITY	ALPHA=
Specifies the c value for the lower bound on risk set size	PROC HPSEVERITY	C=
Specifies the absolute lower bound on risk set size	PROC HPSEVERITY	RSLB=
EMPIRICALCDF=TURNBULL Options		
Specifies that the final EDF estimates be maximum likelihood estimates	PROC HPSEVERITY	ENSUREMLE
Specifies the relative convergence criterion	PROC HPSEVERITY	EPS=
Specifies the maximum number of iterations	PROC HPSEVERITY	MAXITER=
Specifies the threshold below which an EDF estimate is deemed to be 0	PROC HPSEVERITY	ZEROPROB=
Displayed Output Options		
Specifies that distributions be listed to the log without estimating any models that use them	DIST	LISTONLY
Suppresses all displayed output	PROC HPSEVERITY	NOPRINT
Specifies which output to display	PROC HPSEVERITY	PRINT=
Specifies the verbosity of messages printed to the log	PROC HPSEVERITY	VERBOSE=
Specifies that distributions be validated without estimating any models that use them	DIST	VALIDATEONLY

PROC HPSEVERITY Statement

PROC HPSEVERITY *options* **;**

The PROC HPSEVERITY statement invokes the procedure. You can specify two types of *options* in the PROC HPSEVERITY statement. One set of *options* controls input and output. The other set of *options* controls the model estimation and selection process.

The following *options* control the input data sets used by PROC HPSEVERITY and various forms of output generated by PROC HPSEVERITY. The *options* are listed in alphabetical order:

COVOUT

specifies that the OUTEST= data set contain the estimate of the covariance structure of the parameters. This option has no effect if the OUTEST= option is not specified. For more information about how the covariance is reported in the OUTEST= data set, see the section "OUTEST= Data Set" on page 213.

DATA=*SAS-data-set*

names the input data set. If the DATA= option is not specified, then the most recently created SAS data set is used.

INEST=*SAS-data-set*

names the input data set that contains the initial values of the parameter estimates to start the optimization process. The initial values specified in the INIT= option in the DIST statement take precedence over any initial values specified in this data set. For more information about the variables in this data set, see the section "INEST= Data Set" on page 212.

INITSAMPLE (*initsample-option***)**

INITSAMPLE (*initsample-option . . . initsample-option***)**

specifies that a sample of the input data be used for initializing the distribution parameters. If you specify more than one *initsample-option*, then separate them with spaces.

When you do not specify initial values for the distribution parameters, PROC HPSEVERITY needs to compute the empirical distribution function (EDF) estimates as part of the default method for parameter initialization. The EDF estimation process can be expensive, especially when you specify censoring or truncation effects for the loss variable. Furthermore, it is not amenable to parallelism due to the sequential nature of the algorithm for truncation effects. You can use the INITSAMPLE option to specify that only a fraction of the input data be used in order to reduce the time taken to compute the EDF estimates. PROC HPSEVERITY uses the uniform random sampling method to select the sample, the size and randomness of which is controlled by the following *initsample-options*:

FRACTION=*number*

specifies the fraction, between 0 and 1, of the input data to be used for sampling.

SEED=*number*

specifies the seed to be used for the uniform random number generator. This option enables you to select the same sample from the same input data across different runs of PROC HPSEVERITY, which can be useful for replicating the results across different runs. If you do not specify the seed value, PROC HPSEVERITY generates a seed that is based on the system clock.

SIZE=*number*

specifies the size of the sample. If the data are distributed across different nodes, then this size applies to the sample that is prepared at each node. For example, let the input data set of size 100,000 observations be distributed across 10 nodes such that each node has 10,000 observations. If you specify SIZE=1000, then each node computes a local EDF estimate by using a sample of size 1,000 selected randomly from its 10,000 observations. If you specify both of the SIZE= and FRACTION= options, then the value specified in the SIZE= option is used and the FRACTION= option is ignored.

If you do not specify the INITSAMPLE option, then a uniform random sample of at most 10,000 observations is used for EDF estimation.

NOPRINT

turns off all displayed output. If specified, any value specified for the PRINT= option is ignored.

OUTEST=*SAS-data-set*

names the output data set to contain estimates of the parameter values and their standard errors for each model whose parameter estimation process converges. For more information about the variables in this data set, see the section "OUTEST= Data Set" on page 213.

OUTMODELINFO=*SAS-data-set*

names the output data set to contain the status of each fitted model. The status information includes the convergence status of the optimization process that is used to estimate the parameters, the status of estimating the covariance matrix, and whether a model is the best according to the specified selection criterion. For more information about the variables in this data set, see the section "OUTMODELINFO= Data Set" on page 214.

OUTSTAT=*SAS-data-set*

names the output data set to contain the values of statistics of fit for each model whose parameter estimation process converges. For more information about the variables in this data set, see the section "OUTSTAT= Data Set" on page 214.

PRINT < **(***global-display-option***)** > < **=***display-option* >

PRINT < **(***global-display-option***)** > < **=** **(***display-option . . . display-option***)** >

specifies the desired displayed output. If you specify more than one *display-option*, then separate them with spaces and enclose them in parentheses.

The following *global-display-option* is available:

ONLY

turns off the default displayed output and displays only the requested output.

The following *display-options* are available:

ALL

displays all the output.

ALLFITSTATS

displays the comparison of all the statistics of fit for all the models in one table. The table does not include the models whose parameter estimation process does not converge.

CONVSTATUS
displays the convergence status of the parameter estimation process.

DESCSTATS
displays the descriptive statistics for the response variable and the regressor variables, if they are specified.

DISTINFO
displays the information about each specified distribution. For each distribution, the information includes the name, description, validity status, and number of distribution parameters.

ESTIMATES
PARMEST
displays the final estimates of parameters. The estimates are not displayed for models whose parameter estimation process does not converge.

ESTIMATIONDETAILS
displays the details of the estimation process for all the models in one table.

INITIALVALUES
displays the initial values and bounds used for estimating each model.

NLOHISTORY
displays the iteration history of the nonlinear optimization process used for estimating the parameters.

NLOSUMMARY
displays the summary of the nonlinear optimization process used for estimating the parameters.

NONE
displays none of the output. If specified, this option overrides all the other display options. The default displayed output is also suppressed.

SELECTION
SELECT
displays the model selection table.

STATISTICS
FITSTATS
displays the statistics of fit for each model. The statistics of fit are not displayed for models whose parameter estimation process does not converge.

If the PRINT= option is not specified or the ONLY *global-display-option* is not specified, then the default displayed output is equivalent to specifying PRINT=(SELECTION CONVSTATUS NLOSUMMARY STATISTICS ESTIMATES).

VARDEF=*option*
specifies the denominator to use for computing the covariance estimates. You can specify one of the following values for *option*:

DF

specifies that the number of nonmissing observations minus the model degrees of freedom (number of parameters) be used.

N

specifies that the number of nonmissing observations be used.

For more information about the covariance estimation, see the section "Estimating Covariance and Standard Errors" on page 171.

VERBOSE=*verbosity-level*

specifies the amount of messages printed to the SAS log by PROC HPSEVERITY. A higher number prints messages with the same or more detail.

The following *options* control the model estimation and selection process:

CRITERION=*criterion-option*

CRITERIA=*criterion-option*

CRIT=*criterion-option*

specifies the model selection criterion.

If two or more models are specified for estimation, then the one with the best value for the selection criterion is chosen as the best model. If the OUTMODELINFO= data set is specified, then the best model's observation has a value of 1 for the _SELECTED_ variable. You can specify one of the following *criterion-options*:

AD

specifies the Anderson-Darling (AD) statistic value, which is computed by using the empirical distribution function (EDF) estimate, as the selection criterion. A lower value is deemed better.

AIC

specifies Akaike's information criterion (AIC) as the selection criterion. A lower value is deemed better.

AICC

specifies the finite-sample corrected Akaike's information criterion (AICC) as the selection criterion. A lower value is deemed better.

BIC

specifies the Schwarz Bayesian information criterion (BIC) as the selection criterion. A lower value is deemed better.

CUSTOM

specifies the custom objective function as the selection criterion. You can specify this only if you also specify the OBJECTIVE= option. A lower value is deemed better.

CVM

specifies the Craḿer-von Mises (CvM) statistic value, which is computed by using the empirical distribution function (EDF) estimate, as the selection criterion. A lower value is deemed better.

KS

specifies the Kolmogorov-Smirnov (KS) statistic value, which is computed by using the empirical distribution function (EDF) estimate, as the selection criterion. A lower value is deemed better.

LOGLIKELIHOOD

LL

specifies $-2 * \log(L)$ as the selection criterion, where L is the likelihood of the data. A lower value is deemed better. This is the default.

For more information about these *criterion-options*, see the section "Statistics of Fit" on page 185.

EMPIRICALCDF | EDF=*method*

specifies the method to use for computing the nonparametric or empirical estimate of the cumulative distribution function of the data. You can specify one of the following values for *method*:

AUTOMATIC | AUTO

specifies that the method be chosen automatically based on the data specification.

If you do not specify any censoring or truncation, then the standard empirical estimation method (STANDARD) is chosen. If you specify both right-censoring and left-censoring, then Turnbull's estimation method (TURNBULL) is chosen. For all other combinations of censoring and truncation, the Kaplan-Meier method (KAPLANMEIER) is chosen.

KAPLANMEIER | KM

specifies that the product limit estimator proposed by Kaplan and Meier (1958) be used. Specification of this method has no effect when both right-censoring and left-censoring are specified.

MODIFIEDKM | MKM <(*options*)>

specifies that the modified product limit estimator be used. Specification of this method has no effect when both right-censoring and left-censoring are specified.

This method allows Kaplan-Meier's product limit estimates to be more robust by ignoring the contributions to the estimate due to small risk-set sizes. The risk set is the set of observations at the risk of failing, where an observation is said to fail if it has not been processed yet and might experience censoring or truncation. The minimum risk-set size that makes it eligible to be included in the estimation can be specified either as an absolute lower bound on the size (RSLB= option) or a relative lower bound determined by the formula cn^α proposed by Lai and Ying (1991). Values of c and α can be specified by using the C= and ALPHA= options, respectively. By default, the relative lower bound is used with values of $c = 1$ and $\alpha = 0.5$. However, you can modify the default by using the following *options*:

ALPHA | A=*number*

specifies the value to use for α when the lower bound on the risk set size is defined as cn^α. This value must satisfy $0 < \alpha < 1$.

C=*number*

specifies the value to use for c when the lower bound on the risk set size is defined as cn^α. This value must satisfy $c > 0$.

RSLB=*number*
specifies the absolute lower bound on the risk set size to be included in the estimate.

NOTURNBULL
specifies that the method be chosen automatically based on the data specification and that Turnbull's method not be used. This is the default.

This method first replaces each left-censored or interval-censored observation with an uncensored observation. If the resulting set of observations has any truncated or right-censored observations, then the Kaplan-Meier method (KAPLANMEIER) is chosen. Otherwise, the standard empirical estimation method (STANDARD) is chosen. The observations are modified only for the purpose of computing the EDF estimates; the modification does not affect the parameter estimation process.

STANDARD | STD
specifies that the standard empirical estimation method be used. This ignores any censoring or truncation information even if specified, and can thus result in estimates that are more biased than those obtained with other methods more suitable for such data. Specification of this method has no effect when both right-censoring and left-censoring are specified.

TURNBULL | EM <(*options***)>** (Experimental)
specifies that the Turnbull's method be used. An iterative expectation-maximization (EM) algorithm proposed by Turnbull (1976) is used to compute the empirical estimates. If truncation is also specified, then the modification suggested by Frydman (1994) is used.

This method is used if both right-censoring and left-censoring are specified and if you have explicitly specified the EMPIRICALCDF=TURNBULL option.

You can modify the default behavior of the EM algorithm by using the following *options*:

ENSUREMLE
specifies that the final EDF estimates be maximum likelihood estimates. The Kuhn-Tucker conditions are computed for the likelihood maximization problem and checked to ensure that EM algorithm converges to maximum likelihood estimates. The method generalizes the method proposed by Gentleman and Geyer (1994) by taking into account the truncation information, if specified.

EPS=*number*
specifies the maximum relative error to be allowed between estimates of two consecutive iterations. This criterion is used to check the convergence of the algorithm. If you do not specify this option, then PROC HPSEVERITY uses a default value of 1.0E–8.

MAXITER=*number*
specifies the maximum number of iterations to attempt to find the empirical estimates. If you do not specify this option, then PROC HPSEVERITY uses a default value of 500.

ZEROPROB=*number*
specifies the threshold below which an empirical estimate of the probability is considered zero. This option is used to decide if the final estimate is a maximum likelihood estimate. This option does not have an effect if you do not specify the ENSUREMLE option. If you specify the ENSUREMLE option, but do not specify this option, then PROC HPSEVERITY uses a default value of 1.0E–8.

For more information about each of the methods, see the section "Empirical Distribution Function Estimation Methods" on page 179.

OBJECTIVE=*symbol-name* (Experimental)

names the symbol that represents the objective function in the specified SAS programming statements. For each model to be estimated, PROC HPSEVERITY executes the programming statements to compute the value of this symbol for each observation. The values are added across all observations to obtain the value of the objective function. The optimization algorithm estimates the model parameters such that the objective function value is *minimized*. A separate optimization problem is solved for each candidate distribution.

For more information about writing SAS programming statements to define your own objective function, see the section "Custom Objective Functions (Experimental)" on page 209.

DIST Statement

DIST *distribution-name-or-keyword* < *(distribution-option)* > < *distribution-name-or-keyword* < *(distribution-option)* > > ... **;**

The DIST statement specifies candidate distributions to be estimated by the HPSEVERITY procedure. You can specify multiple DIST statements, and each statement can contain one or more distribution specifications.

For your convenience, PROC HPSEVERITY provides the following 10 different predefined distributions (the name in the parentheses is the name to use in the DIST statement): Burr (BURR), exponential (EXP), gamma (GAMMA), generalized Pareto (GPD), inverse Gaussian or Wald (IGAUSS), lognormal (LOGN), Pareto (PARETO), Tweedie (TWEEDIE), scaled Tweedie (STWEEDIE), and Weibull (WEIBULL). These are described in detail in the section "Predefined Distributions" on page 159.

You can specify any of the predefined distributions or any distribution that you have defined. If the specified distribution is not a predefined distribution, then you must submit the CMPLIB= system option with appropriate libraries before you submit the PROC HPSEVERITY step to enable the procedure to find the functions associated with your distribution. The predefined distributions are defined in the Sashelp.Svrtdist library. However, you are not required to specify this library in the CMPLIB= system option.

As a convenience, you can also use a shortcut keyword to indicate a list of distributions. You can specify one or more of the following keywords:

ALL

specifies all the predefined distributions and the distributions that you have defined in the libraries that are specified in the CMPLIB= system option. In addition to the eight predefined distributions included by the _PREDEFINED_ keyword, this list also includes the Tweedie and scaled Tweedie distributions that are defined in the Sashelp.Svrtdist library.

PREDEFINED

specifies the list of eight predefined distributions: BURR, EXP, GAMMA, GPD, IGAUSS, LOGN, PARETO, and WEIBULL. Although the TWEEDIE and STWEEDIE distributions are available in the Sashelp.Svrtdist library along with these eight distributions, they are not included by this keyword. If you want to fit the TWEEDIE and STWEEDIE distributions, then you must specify them explicitly or use the _ALL_ keyword.

USER

specifies the list of all the distributions that you have defined in the libraries that are specified in the CMPLIB= system option. This list does not include the distributions defined in the Sashelp.Svrtdist library, even if you have specified Sashelp.Svrtdist in the CMPLIB= option.

The use of these keywords, especially _ALL_, can result in a large list of distributions, which might take a longer time to estimate. A warning is printed to the SAS log if the number of total distribution models to estimate exceeds 10.

The following *distribution-option* values can be used in the DIST statement for a distribution name that is not a shortcut keyword:

INIT=*(name=value . . . name=value)*

specifies the initial values to be used for the distribution parameters to start the parameter estimation process. The values must be specified by parameter names. The parameter names must match the names used in the model definition. For example, let a model M's definition contain a M_PDF function with following signature:

```
function M_PDF(x, alpha, beta);
```

For this model, the names **`alpha`** and **`beta`** must be used for the INIT option. The names are case-insensitive. If you do not specify initial values for some parameters in the INIT statement, then a default value of 0.001 is assumed for those parameters. If you specify an incorrect parameter, PROC HPSEVERITY prints a warning to the SAS log and does not fit the model. All specified values must be nonmissing.

If you are modeling regression effects, then the initial value of the first distribution parameter (**`alpha`** in the preceding example) should be the initial *base* value of the scale parameter or log-transformed scale parameter. For more information, see the section "Estimating Regression Effects" on page 175.

The use of INIT= option is one of the three methods available for initializing the parameters. For more information, see the section "Parameter Initialization" on page 174. If none of the initialization methods is used, then PROC HPSEVERITY initializes all parameters to 0.001.

You can specify the following *preprocess-options* in the DIST statement:

LISTONLY

specifies that the list of all candidate distributions be printed to the SAS log without doing any further processing on them. This option is especially useful when you use a shortcut keyword to include a list of distributions. It enables you to find out which distributions are included by the keyword.

VALIDATEONLY

specifies that all candidate distributions be checked for validity without doing any further processing on them. If a distribution is invalid, the reason for invalidity is written to the SAS log. If all distributions are valid, then the distribution information is written to the SAS log. The information includes name, description, validity status (valid or invalid), and number of distribution parameters. The information is not written to the SAS log if you have specified an OUTMODELINFO= data set or the PRINT=DISTINFO or PRINT=ALL option in the PROC HPSEVERITY statement. This option is especially useful when you specify your own distributions or when you specify the _USER_ or _ALL_ keywords in the DIST statement. It enables you to check whether your custom distribution definitions satisfy PROC HPSEVERITY's requirements for the specified modeling task. It is recommended that you specify the SCALEMODEL statement if you intend to fit a model with regression effects, because the SCALEMODEL statement instructs PROC HPSEVERITY to perform additional checks to validate whether regression effects can be modeled on each candidate distribution.

LOSS Statement

LOSS < *response-variable-name* > < / *censoring-truncation-options* > **;**

The LOSS statement specifies the name of the response or loss variable whose distribution needs to be modeled. You can also specify additional options to indicate any truncation or censoring of the response. The specification of response variable is optional if at least one type of censoring is specified. You must specify a response variable if no censoring is specified. If you specify more than one LOSS statement, then the first statement is used.

All the analysis variables specified in this statement must be present in the input data set that is specified by using the DATA= option in the PROC HPSEVERITY statement. The response variable is expected to have nonmissing values. If the variable has a missing value in an observation, then a warning is written to the SAS log and that observation is ignored.

The following *censoring-truncation-options* can be used in the LOSS statement:

LEFTCENSORED=*variable-name*
LC=*variable-name*
LEFTCENSORED=*number*
LC=*number*

specifies the left-censoring variable or a global left-censoring limit.

You can use the *variable-name* argument to specify a data set variable that contains the left-censoring limit. If the value of this variable is missing, then PROC HPSEVERITY assumes that such observations are not left-censored.

Alternatively, you can use the *number* argument to specify a left-censoring limit value that applies to all the observations in the data set. This limit must be a nonzero positive number.

By the definition of left-censoring, an exact value of the response is not known when it is less than or equal to the left-censoring limit. If the response variable is specified and the value of that variable is less than or equal to the value of the left-censoring limit for some observations, then PROC HPSEVERITY treats such observations as left-censored and the value of the response variable is ignored. If the response variable is specified and the value of that variable is greater than the value of the left-censoring limit for some observations, then PROC HPSEVERITY assumes that such observations are not left-censored.

If both right-censoring and left-censoring limits are specified, then the left-censoring limit must be greater than or equal to the right-censoring limit. If both limits are identical, then the observation is assumed to be uncensored.

For more information about left-censoring, see the section "Censoring and Truncation" on page 169.

LEFTTRUNCATED | LT=*variable-name*

LT=*variable-name*

LEFTTRUNCATED=*number*

LT=*number*

specifies the left-truncation variable or a global left-truncation threshold.

You can use the *variable-name* argument to specify a data set variable that contains the left-truncation threshold. If the value of this variable is missing or 0 for some observations, then PROC HPSEVERITY assumes that such observations are not left-truncated.

Alternatively, you can use the *number* argument to specify a left-truncation threshold that applies to all the observations in the data set. This threshold must be a nonzero positive number.

It is assumed that the response variable contains the observed values. By the definition of left-truncation, you can observe only a value that is greater than the left-truncation threshold. If a response variable value is less than or equal to the left-truncation threshold, a warning is printed to the SAS log, and the observation is ignored. For more information about left-truncation, see the section "Censoring and Truncation" on page 169.

RIGHTCENSORED | RC=*variable-name*

RC=*variable-name*

RIGHTCENSORED=*number*

RC=*number*

specifies the right-censoring variable or a global right-censoring limit.

You can use the *variable-name* argument to specify a data set variable that contains the right-censoring limit. If the value of this variable is missing, then PROC HPSEVERITY assumes that such observations are not right-censored.

Alternatively, you can use the *number* argument to specify a right-censoring limit value that applies to all the observations in the data set. This limit must be a nonzero positive number.

By the definition of right-censoring, an exact value of the response is not known when it is greater than or equal to the right-censoring limit. If the response variable is specified and the value of that variable is greater than or equal to the value of the right-censoring limit for some observations, then PROC HPSEVERITY treats such observations as right-censored and the value of the response variable is ignored. If the response variable is specified and the value of that variable is less than the value of the right-censoring limit for some observations, then PROC HPSEVERITY assumes that such observations are not right-censored and the value of the right-censoring limit is ignored.

If both right-censoring and left-censoring limits are specified, then the left-censoring limit must be greater than or equal to the right-censoring limit. If both limits are identical, then the observation is assumed to be uncensored.

For more information about right-censoring, see the section "Censoring and Truncation" on page 169.

RIGHTTRUNCATED | RT=*variable-name*

RT=*variable-name*

RIGHTTRUNCATED=*number*

RT=*number*

specifies the right-truncation variable or a global right-truncation threshold.

You can use the *variable-name* argument to specify a data set variable that contains the right-truncation threshold. If the value of this variable is missing for some observations, then PROC HPSEVERITY assumes that such observations are not right-truncated.

Alternatively, you can use the *number* argument to specify a right-truncation threshold that applies to all the observations in the data set. This threshold must be a nonzero positive number.

It is assumed that the response variable contains the observed values. By the definition of right-truncation, you can observe only a value that is less than or equal to the right-truncation threshold. If a response variable value is greater than the right-truncation threshold, a warning is printed to the SAS log, and the observation is ignored. For more information about right-truncation, see the section "Censoring and Truncation" on page 169.

NLOPTIONS Statement

NLOPTIONS *options* **;**

The HPSEVERITY procedure uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization of the likelihood function to obtain the estimates of distribution and regression parameters. You can use the NLOPTIONS statement to control different aspects of this optimization process. If you specify more than one NLOPTIONS statement, then the first statement is used.

For most problems, the default settings of the optimization process are adequate. However, in some cases it might be useful to change the optimization technique or to change the maximum number of iterations. The following statement uses the MAXITER= option to set the maximum number of iterations to 200 and uses the TECH= option to change the optimization technique to the double-dogleg optimization (DBLDOG) rather than the default technique, the trust region optimization (TRUREG), used in the HPSEVERITY procedure:

```
   nloptions tech=dbldog maxiter=200;
```

The following *options* values can be used in the NLOPTIONS statement:

ABSCONV=*r*

ABSTOL=*r*

specifies an absolute function value convergence criterion. For minimization, termination requires $f(\theta^{(k)}) \leq r$. The default value of *r* is the negative square root of the largest double-precision value, which serves only as a protection against overflows.

ABSFCONV=*r*

ABSFTOL=*r*

specifies an absolute function difference convergence criterion. For all techniques except NMSIMP, termination requires a small change of the function value in successive iterations:

$$|f(\theta^{(k-1)}) - f(\theta^{(k)})| \leq r$$

The same formula is used for the NMSIMP technique, but $\theta^{(k)}$ is defined as the vertex with the lowest function value, and $\theta^{(k-1)}$ is defined as the vertex with the highest function value in the simplex. The default value is $r = 0$.

ABSGCONV=*r*

ABSGTOL=*r*

specifies an absolute gradient convergence criterion. Termination requires the maximum absolute gradient element to be small:

$$\max_j |g_j(\theta^{(k)})| \leq r$$

This criterion is not used by the NMSIMP technique. The default value is r=1E–5.

ABSXCONV=*r*

ABSXTOL=*r*

specifies an absolute parameter convergence criterion. For all techniques except NMSIMP, termination requires a small Euclidean distance between successive parameter vectors,

$$\parallel \theta^{(k)} - \theta^{(k-1)} \parallel_2 \leq r$$

For the NMSIMP technique, termination requires either a small length $\alpha^{(k)}$ of the vertices of a restart simplex,

$$\alpha^{(k)} \leq r$$

or a small simplex size,

$$\delta^{(k)} \leq r$$

where the simplex size $\delta^{(k)}$ is defined as the L1 distance from the simplex vertex $\xi^{(k)}$ with the smallest function value to the other simplex points $\theta_l^{(k)} \neq \xi^{(k)}$:

$$\delta^{(k)} = \sum \parallel \theta_l^{(k)} - \xi^{(k)} \parallel_1$$

The default is r=1E–8 for the NMSIMP technique and $r = 0$ otherwise.

FCONV=*r*

FTOL=*r*

specifies a relative function convergence criterion. For all techniques except NMSIMP, termination requires a small relative change of the function value in successive iterations,

$$\frac{|f(\theta^{(k)}) - f(\theta^{(k-1)})|}{\max(|f(\theta^{(k-1)})|, \text{FSIZE})} \leq r$$

where FSIZE is defined by the FSIZE= option. The same formula is used for the NMSIMP technique, but $\theta^{(k)}$ is defined as the vertex with the lowest function value, and $\theta^{(k-1)}$ is defined as the vertex with the highest function value in the simplex.

The default value is $r = 2\epsilon$, where ϵ denotes the machine precision constant, which is the smallest double-precision floating-point number such that $1 + \epsilon > 1$.

FCONV2=*r*

FTOL2=*r*

specifies another function convergence criterion.

For all techniques except NMSIMP, termination requires a small predicted reduction of the objective function:

$$df^{(k)} \approx f(\theta^{(k)}) - f(\theta^{(k)} + s^{(k)})$$

The predicted reduction

$$\begin{aligned} df^{(k)} &= -g^{(k)T}s^{(k)} - \frac{1}{2}s^{(k)T}H^{(k)}s^{(k)} \\ &= -\frac{1}{2}s^{(k)T}g^{(k)} \\ &\leq r \end{aligned}$$

is computed by approximating the objective function f by the first two terms of the Taylor series and substituting the Newton step

$$s^{(k)} = -[H^{(k)}]^{-1}g^{(k)}$$

For the NMSIMP technique, termination requires a small standard deviation of the function values of the $p+1$ simplex vertices $\theta_l^{(k)}$, $l = 0, \ldots, p$, $\sqrt{\frac{1}{p+1}\sum_l \left[f(\theta_l^{(k)}) - \overline{f}(\theta^{(k)})\right]^2} \leq r$ where $\overline{f}(\theta^{(k)}) = \frac{1}{p+1}\sum_l f(\theta_l^{(k)})$. If there are p_{act} boundary constraints active at $\theta^{(k)}$, the mean and standard deviation are computed only for the $p + 1 - p_{act}$ unconstrained vertices.

The default value is *r*=1E–6 for the NMSIMP technique and $r = 0$ otherwise.

FSIZE=*r*

specifies the FSIZE parameter of the relative function and relative gradient termination criteria. The default value is $r = 0$. For more information, see the FCONV= and GCONV= options.

GCONV=*r*

GTOL=*r*

specifies a relative gradient convergence criterion. For all techniques except CONGRA and NMSIMP, termination requires that the normalized predicted function reduction is small,

$$\frac{g(\theta^{(k)})^T[H^{(k)}]^{-1}g(\theta^{(k)})}{\max(|f(\theta^{(k)})|, \text{FSIZE})} \leq r$$

where FSIZE is defined by the FSIZE= option. For the CONGRA technique (where a reliable Hessian estimate H is not available), the following criterion is used:

$$\frac{\| g(\theta^{(k)}) \|_2^2 \quad \| s(\theta^{(k)}) \|_2}{\| g(\theta^{(k)}) - g(\theta^{(k-1)}) \|_2 \max(|f(\theta^{(k)})|, \text{FSIZE})} \leq r$$

This criterion is not used by the NMSIMP technique. The default value is $r = 1E-8$.

MAXFUNC=*i*

MAXFU=*i*

specifies the maximum number *i* of function calls in the optimization process. The default values are

- TRUREG, NRRIDG, NEWRAP: 125
- QUANEW, DBLDOG: 500
- CONGRA: 1000
- NMSIMP: 3000

Note that the optimization can terminate only after completing a full iteration. Therefore, the number of function calls that is actually performed can exceed the number that is specified by the MAXFUNC= option.

MAXITER=*i*

MAXIT=*i*

specifies the maximum number *i* of iterations in the optimization process. The default values are

- TRUREG, NRRIDG, NEWRAP: 50
- QUANEW, DBLDOG: 200
- CONGRA: 400
- NMSIMP: 1000

These default values are also valid when *i* is specified as a missing value.

MAXTIME=*r*

specifies an upper limit of *r* seconds of CPU time for the optimization process. The default value is the largest floating-point double representation of your computer. The time specified by the MAXTIME= option is checked only once at the end of each iteration. Therefore, the actual running time can be much longer than that specified by the MAXTIME= option. The actual running time includes the rest of the time needed to finish the iteration and the time needed to generate the output of the results.

MINITER=*i*

MINIT=*i*

specifies the minimum number of iterations. The default value is 0. If you request more iterations than are actually needed for convergence to a stationary point, the optimization algorithms can behave strangely. For example, the effect of rounding errors can prevent the algorithm from continuing for the required number of iterations.

TECHNIQUE=*name*

TECH=*name*

specifies the optimization technique. Valid values for *name* are as follows:

CONGRA	performs a conjugate-gradient optimization.
DBLDOG	performs a version of double-dogleg optimization.
NMSIMP	performs a Nelder-Mead simplex optimization.

NONE	does not perform any optimization. This option can be used as follows: • to perform a grid search without optimization • to compute estimates and predictions that cannot be obtained efficiently with any of the optimization techniques
NEWRAP	performs a Newton-Raphson optimization that combines a line-search algorithm with ridging.
NRRIDG	performs a Newton-Raphson optimization with ridging.
QUANEW	performs a quasi-Newton optimization.
TRUREG	performs a trust region optimization. This is the default estimation method.

For more information about optimization algorithms, see the section "Details of Optimization Algorithms" on page 172.

XCONV=*r*

XTOL=*r*

specifies the relative parameter convergence criterion. For all techniques except NMSIMP, termination requires a small relative parameter change in subsequent iterations:

$$\frac{\max_j |\theta_j^{(k)} - \theta_j^{(k-1)}|}{\max(|\theta_j^{(k)}|, |\theta_j^{(k-1)}|, \text{XSIZE})} \leq r$$

For the NMSIMP technique, the same formula is used, but $\theta_j^{(k)}$ is defined as the vertex that has the lowest function value and $\theta_j^{(k-1)}$ is defined as the vertex that has the highest function value in the simplex. The default value is r=1E–8 for the NMSIMP technique and $r = 0$ otherwise.

XSIZE=*r*

specifies the XSIZE parameter of the relative parameter termination criterion. The value of r must be greater than or equal to 0; the default is $r = 0$. For more information, see the XCONV= option.

PERFORMANCE Statement

PERFORMANCE *options* **;**

The PERFORMANCE statement defines performance parameters for distributed and multithreaded computing, passes variables that describe the distributed computing environment, and requests detailed results about the performance characteristics of PROC HPSEVERITY.

You can also use the PERFORMANCE statement to control whether a high-performance analytical procedure executes in single-machine or distributed mode.

The PERFORMANCE statement is documented further in the section "PERFORMANCE Statement" on page 34 of Chapter 2, "Shared Concepts and Topics."

SCALEMODEL Statement

SCALEMODEL *regressor-variable-list* < / *scalemodel-option* > **;**

The SCALEMODEL statement specifies regression variables. All the variables specified in this statement must be present in the input data set that is specified by the DATA= option in the PROC HPSEVERITY statement. The scale parameter of each candidate distribution is linked to a linear combination of these regression variables along with an intercept. If a distribution does not have a scale parameter, then a model based on that distribution is not estimated. If you specify more than one SCALEMODEL statement, then the first statement is used.

The regressor variables are expected to have nonmissing values. If any of the variables has a missing value in an observation, then a warning is written to the SAS log and that observation is ignored.

For more information about modeling regression effects, see the section "Estimating Regression Effects" on page 175.

You can specify the following *scalemodel-option* in the SCALEMODEL statement:

DFMIXTURE=*method-name* < *(method-options)* >

specifies the method for computing representative estimates of the cumulative distribution function (CDF).

When regression variables are specified, the scale of the distribution depends on the values of the regressors. For a given distribution family, each observation in the input data set implies a different scaled version of the distribution. To compute estimates of CDF that are comparable across different distribution families, PROC HPSEVERITY needs to construct a single representative distribution from all such distributions. You can specify one of the following *method-name* values to specify the method that is used to construct the representative distribution. For more information about each of the methods, see the section "CDF Estimates with Regression Effects" on page 177.

FULL

specifies that the representative distribution be the mixture of N distributions such that each distribution has a scale value that is implied by each of the N observations that are used for estimation. This method is the slowest.

MEAN

specifies that the representative distribution be the one-point mixture of the distribution whose scale value is the mean of the N scale values that are implied by the N observations that are used for estimation. If you do not specify the DFMIXTURE= option, then this method is used by default. This is also the fastest method.

QUANTILE < **(K=***q***)** >

specifies that the representative distribution be the mixture of a fixed number of distributions whose scale values are the quantiles from the sample of N scale values that are implied by the N observations in the current BY group (or in the entire DATA= data set if the BY statement is not specified).

You can use the K= option to specify the number of distributions in the mixture. If you specify K=*q*, then the mixture contains $(q - 1)$ distributions such that each distribution has as its scale one of the $(q - 1)$-quantiles.

If you do not specify the K= option, then PROC HPSEVERITY uses the default of 2, which implies the use of a one-point mixture with a distribution whose scale value is the median of all scale values.

RANDOM < **(***random-method-options***)** >

specifies that the representative distribution be the mixture of a fixed number of distributions whose scale values are the scale values that are implied by a randomly chosen subset of the set of all observations in the current BY group (or in the entire DATA= data set if the BY statement is not specified). The same subset of observations is used for each distribution family.

You can specify the following *random-method-options* to specify how the subset is chosen:

K=*r*

specifies the number of distributions to include in the mixture. If you do not specify this option, then PROC HPSEVERITY uses the default of 15.

SEED=*number*

specifies the seed that is used to generate the uniform random sample of observation indices. If you do not specify this option, then PROC HPSEVERITY generates a seed internally that is based on the current value of the system clock.

WEIGHT Statement

WEIGHT *variable-name* **;**

The WEIGHT statement specifies the name of a variable whose values represent the weight of each observation. PROC HPSEVERITY associates a weight of w to each observation, where w is the value of the WEIGHT variable for the observation. If the weight value is missing or less than or equal to 0, then the observation is ignored and a warning is written to the SAS log. When the WEIGHT statement is not specified, each observation is assigned a weight of 1. If you specify more than one WEIGHT statement, then the last statement is used.

The weights are normalized so that they add up to the actual sample size. In particular, the weight of each observation is multiplied by $\frac{N}{\sum_{i=1}^{N} w_i}$, where N is the sample size.

Programming Statements (Experimental)

You can use a series of programming statements that use variables in the input data set specified by DATA= option in the PROC HPSEVERITY statement to assign a value to an objective function symbol. The objective function symbol must be specified using the OBJECTIVE= option in the PROC HPSEVERITY statement. If you do not specify the OBJECTIVE= option in the PROC HPSEVERITY statement, then the programming statements are ignored and models are estimated using the maximum likelihood method.

You can use most DATA step statements and functions in your program. Any additional functions, restrictions, and differences are listed in the section "Custom Objective Functions (Experimental)" on page 209.

Details: HPSEVERITY Procedure

Predefined Distributions

PROC HPSEVERITY assumes the following model for the response variable Y

$$Y \sim \mathcal{F}(\Theta)$$

where $\mathcal{F}$ is a continuous probability distribution with parameters Θ. The model hypothesizes that the observed response is generated from a stochastic process that is governed by the distribution $\mathcal{F}$. This model is usually referred to as the error model. Given a representative input sample of response variable values, PROC HPSEVERITY estimates the model parameters for any distribution $\mathcal{F}$ and computes the statistics of fit for each model. This enables you to find the distribution that is most likely to generate the observed sample.

A set of predefined distributions is provided with the HPSEVERITY procedure. A summary of the distributions is provided in Table 5.2. For each distribution, the table lists the name of the distribution that should be used in the DIST statement, the parameters of the distribution along with their bounds, and the mathematical expressions for the probability density function (PDF) and cumulative distribution function (CDF) of the distribution.

All the predefined distributions, except LOGN and TWEEDIE, are parameterized such that their first parameter is the scale parameter. For LOGN, the first parameter μ is a log-transformed scale parameter. TWEEDIE does not have a scale parameter. The presence of scale parameter or a log-transformed scale parameter enables you to use all of the predefined distributions, except TWEEDIE, as a candidate for estimating regression effects.

A distribution model is associated with each predefined distribution. You can also define your own distribution model, which is a set of functions and subroutines that you define by using the FCMP procedure. For more information, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 192.

Table 5.2 Predefined HPSEVERITY Distributions

Name	**Distribution**	**Parameters**	**PDF (f) and CDF (F)**
BURR	Burr	$\theta > 0, \alpha > 0,$ $\gamma > 0$	$f(x) = \frac{\alpha\gamma z^{\gamma}}{x(1+z^{\gamma})^{(\alpha+1)}}$ $F(x) = 1 - \left(\frac{1}{1+z^{\gamma}}\right)^{\alpha}$
EXP	Exponential	$\theta > 0$	$f(x) = \frac{1}{\theta}e^{-z}$ $F(x) = 1 - e^{-z}$
GAMMA	Gamma	$\theta > 0, \alpha > 0$	$f(x) = \frac{z^{\alpha}e^{-z}}{x\Gamma(\alpha)}$ $F(x) = \frac{\gamma(\alpha,z)}{\Gamma(\alpha)}$
GPD	Generalized Pareto	$\theta > 0, \xi > 0$	$f(x) = \frac{1}{\theta}(1+\xi z)^{-1-1/\xi}$ $F(x) = 1 - (1+\xi z)^{-1/\xi}$
IGAUSS	Inverse Gaussian (Wald)	$\theta > 0, \alpha > 0$	$f(x) = \frac{1}{\theta}\sqrt{\frac{\alpha}{2\pi z^3}}\, e^{\frac{-\alpha(z-1)^2}{2z}}$ $F(x) = \Phi\left((z-1)\sqrt{\frac{\alpha}{z}}\right) + \Phi\left(-(z+1)\sqrt{\frac{\alpha}{z}}\right)\, e^{2\alpha}$
LOGN	Lognormal	μ (no bounds), $\sigma > 0$	$f(x) = \frac{1}{x\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{\log(x)-\mu}{\sigma}\right)^2}$ $F(x) = \Phi\left(\frac{\log(x)-\mu}{\sigma}\right)$
PARETO	Pareto	$\theta > 0, \alpha > 0$	$f(x) = \frac{\alpha\theta^{\alpha}}{(x+\theta)^{\alpha+1}}$ $F(x) = 1 - \left(\frac{\theta}{x+\theta}\right)^{\alpha}$
TWEEDIE	Tweedie[6]	$\mu > 0, \phi > 0,$ $p > 1$	$f(x) = a(x,\phi)\exp\left[\frac{1}{\phi}\left(\frac{x\mu^{1-p}}{1-p} - \kappa(\mu,p)\right)\right]$ $F(x) = \int_0^x f(t)dt$
STWEEDIE	Scaled Tweedie[6]	$\theta > 0, \lambda > 0,$ $1 < p < 2$	$f(x) = a(x,\theta,\lambda,p)\exp\left(-\frac{x}{\theta} - \lambda\right)$ $F(x) = \int_0^x f(t)dt$
WEIBULL	Weibull	$\theta > 0, \tau > 0$	$f(x) = \frac{1}{x}\tau z^{\tau}e^{-z^{\tau}}$ $F(x) = 1 - e^{-z^{\tau}}$

Notes:
1. $z = x/\theta$, wherever z is used.
2. θ denotes the scale parameter for all the distributions. For LOGN, $\log(\theta) = \mu$.
3. Parameters are listed in the order in which they are defined in the distribution model.
4. $\gamma(a,b) = \int_0^b t^{a-1}e^{-t}dt$ is the lower incomplete gamma function.
5. $\Phi(y) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{y}{\sqrt{2}}\right)\right)$ is the standard normal CDF.
6. For more information, see the section "Tweedie Distributions" on page 161.

Tweedie Distributions

Tweedie distributions are a special case of the exponential dispersion family (Jørgensen 1987) with a property that the variance of the distribution is equal to $\phi\mu^p$, where μ is the mean of the distribution, ϕ is a dispersion parameter, and p is an index parameter as discovered by Tweedie (1984). The distribution is defined for all values of p except for values of p in the open interval $(0, 1)$. Many important known distributions are a special case of Tweedie distributions including normal (p=0), Poisson (p=1), gamma (p=2), and the inverse Gaussian (p=3). Apart from these special cases, the probability density function (PDF) of the Tweedie distribution does not have an analytic expression. For $p > 1$, it has the form (Dunn and Smyth 2005),

$$f(x;\mu,\phi,p) = a(x,\phi)\exp\left[\frac{1}{\phi}\left(\frac{x\mu^{1-p}}{1-p} - \kappa(\mu,p)\right)\right]$$

where $\kappa(\mu,p) = \mu^{2-p}/(2-p)$ for $p \neq 2$ and $\kappa(\mu,p) = \log(\mu)$ for $p = 2$. The function $a(x,\phi)$ does not have an analytical expression. It is typically evaluated using series expansion methods described in Dunn and Smyth (2005).

For $1 < p < 2$, the Tweedie distribution is a compound Poisson-gamma mixture distribution, which is the distribution of S defined as

$$S = \sum_{i=1}^{N} X_i$$

where $N \sim \text{Poisson}(\lambda)$ and $X_i \sim \text{gamma}(\alpha,\theta)$ are independent and identically distributed gamma random variables with shape parameter α and scale parameter θ. At $X = 0$, the density is a probability mass that is governed by the Poisson distribution, and for values of $X > 0$, it is a mixture of gamma variates with Poisson mixing probability. The parameters λ, α, and θ are related to the natural parameters μ, ϕ, and p of the Tweedie distribution as

$$\begin{aligned}\lambda &= \frac{\mu^{2-p}}{\phi(2-p)}\\ \alpha &= \frac{2-p}{p-1}\\ \theta &= \phi(p-1)\mu^{p-1}\end{aligned}$$

The mean of a Tweedie distribution is positive for $p > 1$.

Two predefined versions of the Tweedie distribution are provided with the HPSEVERITY procedure. The first version, named TWEEDIE and defined for $p > 1$, has the natural parameterization with parameters μ, ϕ, and p. The second version, named STWEEDIE and defined for $1 < p < 2$, is the version with a scale parameter. It corresponds to the compound Poisson-gamma distribution with gamma scale parameter θ, Poisson mean parameter λ, and the index parameter p. The index parameter decides the shape parameter α of the gamma distribution as

$$\alpha = \frac{2-p}{p-1}$$

The parameters θ and λ of the STWEEDIE distribution are related to the parameters μ and ϕ of the TWEEDIE distribution as

$$\begin{aligned}\mu &= \lambda\theta\alpha\\ \phi &= \frac{(\lambda\theta\alpha)^{2-p}}{\lambda(2-p)} = \frac{\theta}{(p-1)(\lambda\theta\alpha)^{p-1}}\end{aligned}$$

You can fit either version when there are no regression variables. Each version has its own merits. If you fit the TWEEDIE version, you have the direct estimate of the overall mean of the distribution. If you are interested in the most practical range of the index parameter $1 < p < 2$, then you can fit the STWEEDIE version, which provides you direct estimates of the Poisson and gamma components that comprise the distribution (an estimate of the gamma shape parameter α is easily obtained from the estimate of p).

If you want to estimate the effect of exogenous (regression) variables on the distribution, then you must use the STWEEDIE version, because PROC HPSEVERITY requires a distribution to have a scale parameter in order to estimate regression effects. For more information, see the section "Estimating Regression Effects" on page 175. The gamma scale parameter θ is the scale parameter of the STWEEDIE distribution. If you are interested in determining the effect of regression variables on the mean of the distribution, you can do so by first fitting the STWEEDIE distribution to determine the effect of the regression variables on the scale parameter θ. Then, you can easily estimate how the mean of the distribution μ is affected by the regression variables using the relationship $\mu = c\theta$, where $c = \lambda\alpha = \lambda(2-p)/(p-1)$. The estimates of the regression parameters remain the same, whereas the estimate of the intercept parameter is adjusted by the estimates of the λ and p parameters.

Parameter Initialization for Predefined Distributions

The parameters are initialized by using the method of moments for all the distributions, except for the gamma and the Weibull distributions. For the gamma distribution, approximate maximum likelihood estimates are used. For the Weibull distribution, the method of percentile matching is used.

Given n observations of the severity value y_i $(1 \le i \le n)$, the estimate of kth raw moment is denoted by m_k and computed as

$$m_k = \frac{1}{n}\sum_{i=1}^{n} y_i^k$$

The $100p$th percentile is denoted by π_p $(0 \le p \le 1)$. By definition, π_p satisfies

$$F(\pi_p -) \le p \le F(\pi_p)$$

where $F(\pi_p -) = \lim_{h \downarrow 0} F(\pi_p - h)$. PROC HPSEVERITY uses the following practical method of computing π_p. Let $\hat{F}_n(y)$ denote the empirical distribution function (EDF) estimate at a severity value y. Let y_p^- and y_p^+ denote two consecutive values in the ascending sequence of y values such that $\hat{F}_n(y_p^-) < p$ and $\hat{F}_n(y_p^+) \ge p$. Then, the estimate $\hat{\pi}_p$ is computed as

$$\hat{\pi}_p = y_p^- + \frac{p - \hat{F}_n(y_p^-)}{\hat{F}_n(y_p^+) - \hat{F}_n(y_p^-)}(y_p^+ - y_p^-)$$

Let ϵ denote the smallest double-precision floating-point number such that $1 + \epsilon > 1$. This machine precision constant can be obtained by using the CONSTANT function in Base SAS software.

The details of how parameters are initialized for each predefined distribution are as follows:

BURR The parameters are initialized by using the method of moments. The kth raw moment of the Burr distribution is:

$$E[X^k] = \frac{\theta^k \Gamma(1 + k/\gamma)\Gamma(\alpha - k/\gamma)}{\Gamma(\alpha)}, \quad -\gamma < k < \alpha\gamma$$

Three moment equations $E[X^k] = m_k$ $(k = 1, 2, 3)$ need to be solved for initializing the three parameters of the distribution. In order to get an approximate closed form solution, the second shape parameter $\hat{\gamma}$ is initialized to a value of 2. If $2m_3 - 3m_1m_2 > 0$, then simplifying and solving the moment equations yields the following feasible set of initial values:

$$\hat{\theta} = \sqrt{\frac{m_2m_3}{2m_3 - 3m_1m_2}}, \quad \hat{\alpha} = 1 + \frac{m_3}{2m_3 - 3m_1m_2}, \quad \hat{\gamma} = 2$$

If $2m_3 - 3m_1m_2 < \epsilon$, then the parameters are initialized as follows:

$$\hat{\theta} = \sqrt{m_2}, \quad \hat{\alpha} = 2, \quad \hat{\gamma} = 2$$

EXP The parameters are initialized by using the method of moments. The kth raw moment of the exponential distribution is:

$$E[X^k] = \theta^k \Gamma(k+1), \quad k > -1$$

Solving $E[X] = m_1$ yields the initial value of $\hat{\theta} = m_1$.

GAMMA The parameter α is initialized by using its *approximate* maximum likelihood (ML) estimate. For a set of n independent and identically distributed observations y_i $(1 \le i \le n)$ drawn from a gamma distribution, the log likelihood l is defined as follows:

$$\begin{aligned} l &= \sum_{i=1}^{n} \log\left(y_i^{\alpha-1} \frac{e^{-y_i/\theta}}{\theta^\alpha \Gamma(\alpha)} \right) \\ &= (\alpha - 1)\sum_{i=1}^{n} \log(y_i) - \frac{1}{\theta}\sum_{i=1}^{n} y_i - n\alpha\log(\theta) - n\log(\Gamma(\alpha)) \end{aligned}$$

Using a shorter notation of $\sum$ to denote $\sum_{i=1}^{n}$ and solving the equation $\partial l / \partial \theta = 0$ yields the following ML estimate of θ:

$$\hat{\theta} = \frac{\sum y_i}{n\alpha} = \frac{m_1}{\alpha}$$

Substituting this estimate in the expression of l and simplifying gives

$$l = (\alpha - 1)\sum \log(y_i) - n\alpha - n\alpha\log(m_1) + n\alpha\log(\alpha) - n\log(\Gamma(\alpha))$$

Let d be defined as follows:

$$d = \log(m_1) - \frac{1}{n}\sum \log(y_i)$$

Solving the equation $\partial l / \partial \alpha = 0$ yields the following expression in terms of the digamma function, $\psi(\alpha)$:

$$\log(\alpha) - \psi(\alpha) = d$$

The digamma function can be approximated as follows:

$$\hat{\psi}(\alpha) \approx \log(\alpha) - \frac{1}{\alpha}\left(0.5 + \frac{1}{12\alpha + 2}\right)$$

This approximation is within 1.4% of the true value for all the values of $\alpha > 0$ except when α is arbitrarily close to the positive root of the digamma function (which is approximately 1.461632). Even for the values of α that are close to the positive root, the absolute error between true and approximate values is still acceptable ($|\hat{\psi}(\alpha) - \psi(\alpha)| < 0.005$ for $\alpha > 1.07$). Solving the equation that arises from this approximation yields the following estimate of α:

$$\hat{\alpha} = \frac{3 - d + \sqrt{(d-3)^2 + 24d}}{12d}$$

If this approximate ML estimate is infeasible, then the method of moments is used. The kth raw moment of the gamma distribution is:

$$E[X^k] = \theta^k \frac{\Gamma(\alpha + k)}{\Gamma(\alpha)}, \quad k > -\alpha$$

Solving $E[X] = m_1$ and $E[X^2] = m_2$ yields the following initial value for α:

$$\hat{\alpha} = \frac{m_1^2}{m_2 - m_1^2}$$

If $m_2 - m_1^2 < \epsilon$ (almost zero sample variance), then α is initialized as follows:

$$\hat{\alpha} = 1$$

After computing the estimate of α, the estimate of θ is computed as follows:

$$\hat{\theta} = \frac{m_1}{\hat{\alpha}}$$

Both the maximum likelihood method and the method of moments arrive at the same relationship between $\hat{\alpha}$ and $\hat{\theta}$.

GPD The parameters are initialized by using the method of moments. Notice that for $\xi > 0$, the CDF of the generalized Pareto distribution (GPD) is:

$$\begin{aligned} F(x) &= 1 - \left(1 + \frac{\xi x}{\theta}\right)^{-1/\xi} \\ &= 1 - \left(\frac{\theta/\xi}{x + \theta/\xi}\right)^{1/\xi} \end{aligned}$$

This is equivalent to a Pareto distribution with scale parameter $\theta_1 = \theta/\xi$ and shape parameter $\alpha = 1/\xi$. Using this relationship, the parameter initialization method used for the PARETO distribution is used to get the following initial values for the parameters of the GPD distribution:

$$\hat{\theta} = \frac{m_1 m_2}{2(m_2 - m_1^2)}, \quad \hat{\xi} = \frac{m_2 - 2m_1^2}{2(m_2 - m_1^2)}$$

If $m_2 - m_1^2 < \epsilon$ (almost zero sample variance) or $m_2 - 2m_1^2 < \epsilon$, then the parameters are initialized as follows:

$$\hat{\theta} = \frac{m_1}{2}, \quad \hat{\xi} = \frac{1}{2}$$

IGAUSS The parameters are initialized by using the method of moments. The standard parameterization of the inverse Gaussian distribution (also known as the Wald distribution), in terms of the location parameter μ and shape parameter λ, is as follows (Klugman, Panjer, and Willmot 1998, p. 583):

$$f(x) = \sqrt{\frac{\lambda}{2\pi x^3}} \exp\left(\frac{-\lambda(x-\mu)^2}{2\mu^2 x}\right)$$
$$F(x) = \Phi\left(\left(\frac{x}{\mu}-1\right)\sqrt{\frac{\lambda}{x}}\right) + \Phi\left(-\left(\frac{x}{\mu}+1\right)\sqrt{\frac{\lambda}{x}}\right) \exp\left(\frac{2\lambda}{\mu}\right)$$

For this parameterization, it is known that the mean is $E[X] = \mu$ and the variance is $Var[X] = \mu^3/\lambda$, which yields the second raw moment as $E[X^2] = \mu^2(1+\mu/\lambda)$ (computed by using $E[X^2] = Var[X] + (E[X])^2$).

The predefined IGAUSS distribution in PROC HPSEVERITY uses the following alternate parameterization to allow the distribution to have a scale parameter, θ:

$$f(x) = \sqrt{\frac{\alpha\theta}{2\pi x^3}} \exp\left(\frac{-\alpha(x-\theta)^2}{2x\theta}\right)$$
$$F(x) = \Phi\left(\left(\frac{x}{\theta}-1\right)\sqrt{\frac{\alpha\theta}{x}}\right) + \Phi\left(-\left(\frac{x}{\theta}+1\right)\sqrt{\frac{\alpha\theta}{x}}\right) \exp(2\alpha)$$

The parameters θ (scale) and α (shape) of this alternate form are related to the parameters μ and λ of the preceding form such that $\theta = \mu$ and $\alpha = \lambda/\mu$. Using this relationship, the first and second raw moments of the IGAUSS distribution are:

$$E[X] = \theta$$
$$E[X^2] = \theta^2\left(1+\frac{1}{\alpha}\right)$$

Solving $E[X] = m_1$ and $E[X^2] = m_2$ yields the following initial values:

$$\hat{\theta} = m_1, \quad \hat{\alpha} = \frac{m_1^2}{m_2 - m_1^2}$$

If $m_2 - m_1^2 < \epsilon$ (almost zero sample variance), then the parameters are initialized as follows:

$$\hat{\theta} = m_1, \quad \hat{\alpha} = 1$$

LOGN The parameters are initialized by using the method of moments. The kth raw moment of the lognormal distribution is:

$$E[X^k] = \exp\left(k\mu + \frac{k^2\sigma^2}{2}\right)$$

Solving $E[X] = m_1$ and $E[X^2] = m_2$ yields the following initial values:

$$\hat{\mu} = 2\log(m1) - \frac{\log(m2)}{2}, \quad \hat{\sigma} = \sqrt{\log(m2) - 2\log(m1)}$$

PARETO The parameters are initialized by using the method of moments. The kth raw moment of the Pareto distribution is:

$$E[X^k] = \frac{\theta^k \Gamma(k+1)\Gamma(\alpha - k)}{\Gamma(\alpha)}, -1 < k < \alpha$$

Solving $E[X] = m_1$ and $E[X^2] = m_2$ yields the following initial values:

$$\hat{\theta} = \frac{m_1 m_2}{m_2 - 2m_1^2}, \quad \hat{\alpha} = \frac{2(m_2 - m_1^2)}{m_2 - 2m_1^2}$$

If $m_2 - m_1^2 < \epsilon$ (almost zero sample variance) or $m_2 - 2m_1^2 < \epsilon$, then the parameters are initialized as follows:

$$\hat{\theta} = m_1, \quad \hat{\alpha} = 2$$

TWEEDIE The parameter p is initialized by assuming that the sample is generated from a gamma distribution with shape parameter α and by computing $\hat{p} = \frac{\hat{\alpha}+2}{\hat{\alpha}+1}$. The initial value $\hat{\alpha}$ is obtained from using the method previously described for the GAMMA distribution. The parameter μ is the mean of the distribution. Hence, it is initialized to the sample mean as

$$\hat{\mu} = m_1$$

Variance of a Tweedie distribution is equal to $\phi\mu^p$. Thus, the sample variance is used to initialize the value of ϕ as

$$\hat{\phi} = \frac{m_2 - m_1^2}{\hat{\mu}^{\hat{p}}}$$

STWEEDIE STWEEDIE is a compound Poisson-gamma mixture distribution with mean $\mu = \lambda\theta\alpha$, where α is the shape parameter of the gamma random variables in the mixture and the parameter p is determined solely by α. First, the parameter p is initialized by assuming that the sample is generated from a gamma distribution with shape parameter α and by computing $\hat{p} = \frac{\hat{\alpha}+2}{\hat{\alpha}+1}$. The initial value $\hat{\alpha}$ is obtained from using the method previously described for the GAMMA distribution. As done for initializing the parameters of the TWEEDIE distribution, the sample mean and variance are used to compute the values $\hat{\mu}$ and $\hat{\phi}$ as

$$\hat{\mu} = m_1$$

$$\hat{\phi} = \frac{m_2 - m_1^2}{\hat{\mu}^{\hat{p}}}$$

Based on the relationship between the parameters of TWEEDIE and STWEEDIE distributions described in the section "Tweedie Distributions" on page 161, values of θ and λ are initialized as

$$\hat{\theta} = \hat{\phi}(\hat{p} - 1)\hat{\mu}^{p-1}$$
$$\hat{\lambda} = \frac{\hat{\mu}}{\hat{\theta}\hat{\alpha}}$$

WEIBULL The parameters are initialized by using the percentile matching method. Let $q1$ and $q3$ denote the estimates of the 25th and 75th percentiles, respectively. Using the formula for the CDF of Weibull distribution, they can be written as

$$1 - \exp(-(q1/\theta)^{\tau}) = 0.25$$
$$1 - \exp(-(q3/\theta)^{\tau}) = 0.75$$

Simplifying and solving these two equations yields the following initial values:

$$\hat{\theta} = \exp\left(\frac{r\log(q1) - \log(q3)}{r - 1}\right), \quad \hat{\tau} = \frac{\log(\log(4))}{\log(q3) - \log(\hat{\theta})}$$

where $r = \log(\log(4))/\log(\log(4/3))$. These initial values agree with those suggested in Klugman, Panjer, and Willmot (1998).

A summary of the initial values of all the parameters for all the predefined distributions is given in Table 5.3. The table also provides the names of the parameters to use in the INIT= option in the DIST statement if you want to provide a different initial value.

Table 5.3 Parameter Initialization for Predefined Distributions

Distribution	Parameter	Name for INIT option	Default Initial Value
BURR	θ	theta	$\sqrt{\frac{m_2 m_3}{2m_3 - 3m_1 m_2}}$
	α	alpha	$1 + \frac{m_3}{2m_3 - 3m_1 m_2}$
	γ	gamma	2
EXP	θ	theta	m_1
GAMMA	θ	theta	m_1/α
	α	alpha	$\frac{3-d+\sqrt{(d-3)^2+24d}}{12d}$
GPD	θ	theta	$m_1 m_2/(2(m_2 - m_1^2))$
	ξ	xi	$(m_2 - 2m_1^2)/(2(m_2 - m_1^2))$
IGAUSS	θ	theta	m_1
	α	alpha	$m_1^2/(m_2 - m_1^2)$
LOGN	μ	mu	$2\log(m1) - \log(m2)/2$
	σ	sigma	$\sqrt{\log(m2) - 2\log(m1)}$
PARETO	θ	theta	$m_1 m_2/(m_2 - 2m_1^2)$
	α	alpha	$2(m_2 - m_1^2)/(m_2 - 2m_1^2)$
TWEEDIE	μ	mu	m_1
	ϕ	phi	$(m_2 - m_1^2)/m_1^p$
	p	p	$(\alpha + 2)/(\alpha + 1)$
			where $\alpha = \frac{3-d+\sqrt{(d-3)^2+24d}}{12d}$
STWEEDIE	θ	theta	$(m_2 - m_1^2)(p-1)/m_1$
	λ	lambda	$m_1^2/(\alpha(m_2 - m_1^2)(p-1))$
	p	p	$(\alpha + 2)/(\alpha + 1)$
			where $\alpha = \frac{3-d+\sqrt{(d-3)^2+24d}}{12d}$
WEIBULL	θ	theta	$\exp\left(\frac{r\log(q1) - \log(q3)}{r-1}\right)$
	τ	tau	$\log(\log(4))/(\log(q3) - \log(\hat{\theta}))$

Notes:

- m_k denotes the kth raw moment
- $d = \log(m_1) - (\sum \log(y_i))/n$
- $q1$ and $q3$ denote the 25th and 75th percentiles, respectively
- $r = \log(\log(4))/\log(\log(4/3))$

Censoring and Truncation

One of the key features of PROC HPSEVERITY is that it enables you to specify whether the severity event's magnitude is observable and if it is observable, then whether the exact value of the magnitude is known. If an event is unobservable when the magnitude is in certain intervals, then it is referred to as a truncation effect. If the exact magnitude of the event is not known, but it is known to have a value in a certain interval, then it is referred to as a censoring effect.

PROC HPSEVERITY allows a severity event to be subject to any combination of the following four censoring and truncation effects:

- Left-truncation: An event is said to be left-truncated if it is observed only when $Y > T^l$, where Y denotes the random variable for the magnitude and T^l denotes a random variable for the truncation threshold. You can specify left-truncation using the LEFTTRUNCATED= option in the LOSS statement.
- Right-truncation: An event is said to be right-truncated if it is observed only when $Y \leq T^r$, where Y denotes the random variable for the magnitude and T^r denotes a random variable for the truncation threshold. You can specify right-truncation using the RIGHTTRUNCATED= option in the LOSS statement.
- Left-censoring: An event is said to be left-censored if it is known that the magnitude is $Y \leq C^l$, but the exact value of Y is not known. C^l is a random variable for the censoring limit. You can specify left-censoring using the LEFTCENSORED= option in the LOSS statement.
- Right-censoring: An event is said to be right-censored if it is known that the magnitude is $Y > C^r$, but the exact value of Y is not known. C^r is a random variable for the censoring limit. You can specify right-censoring using the RIGHTCENSORED= option in the LOSS statement.

For each effect, you can specify a different threshold or limit for each observation or specify a single threshold or limit that applies to all the observations.

If all the four types of effects are present on an event, then the following relationship holds: $T^l < C^r \leq C^l \leq T^r$. PROC HPSEVERITY checks these relationships and write a warning to the SAS log if any is violated.

If the response variable is specified in the LOSS statement, then PROC HPSEVERITY also checks whether each observation satisfies the definitions of the specified censoring and truncation effects. If left-truncation is specified, then PROC HPSEVERITY ignores observations where $Y \leq T^l$, because such observations are not observable by definition. Similarly, if right-truncation is specified, then PROC HPSEVERITY ignores observations where $Y > T^r$. If left-censoring is specified, then PROC HPSEVERITY treats an observation with $Y > C^l$ as uncensored and ignores the value of C^l. The observations with $Y \leq C^l$ are considered as left-censored, and the value of Y is ignored. If right-censoring is specified, then PROC HPSEVERITY treats an observation with $Y \leq C^r$ as uncensored and ignores the value of C^r. The observations with $Y > C^r$ are considered as right-censored, and the value of Y is ignored. The specification of both left-censoring and right-censoring is referred to as interval-censoring. If $C^r < C^l$ is satisfied for an observation, then it is considered as interval-censored and the value of the response variable is ignored. If $C^r = C^l$ for an observation, then PROC HPSEVERITY assumes that observation to be uncensored. If all the observations in a data set are censored in some form, then the specification of the response variable in the LOSS statement is optional, because the actual value of the response variable is not required for the purposes of estimating a model.

Specification of censoring and truncation affects the likelihood of the data (see the section "Likelihood Function" on page 170) and how the empirical distribution function (EDF) is estimated (see the section "Empirical Distribution Function Estimation Methods" on page 179).

Truncation and Conditional CDF Estimates

If left-truncation or right-truncation is specified, then the EDF estimates that are computed by all methods except the STANDARD method are conditional on the truncation information. See the section "EDF Estimates and Truncation" on page 184 for more information. In such cases, PROC HPSEVERITY uses conditional estimates of the CDF when it computes the EDF-based statistics of fit.

Let $t^l_{\text{min}} = \min_i\{t^l_i\}$ be the smallest value of the left-truncation threshold (t^l_i is the left-truncation threshold for observation i) and $t^r_{\text{max}} = \max_i\{t^r_i\}$ be the largest value of the right-truncation threshold (t^r_i is the right-truncation threshold for observation i). If $\hat{F}(y)$ denotes the unconditional estimate of the CDF at y, then the conditional estimate $\hat{F}^c(y)$ is computed as follows:

- If an observation is both left-truncated and right-truncated, then

$$\hat{F}^c(y) = \frac{\hat{F}(y) - \hat{F}(t^l_{\text{min}})}{\hat{F}(t^r_{\text{max}}) - \hat{F}(t^l_{\text{min}})}$$

- If an observation is left-truncated but not right-truncated, then

$$\hat{F}^c(y) = \frac{\hat{F}(y) - \hat{F}(t^l_{\text{min}})}{1 - \hat{F}(t^l_{\text{min}})}$$

- If an observation is right-truncated but not left-truncated, then

$$\hat{F}^c(y) = \frac{\hat{F}(y)}{\hat{F}(t^r_{\text{max}})}$$

If regressors are specified, then $\hat{F}(y)$, $\hat{F}(t^l_{\text{min}})$, and $\hat{F}(t^r_{\text{max}})$ are all computed from a mixture distribution, as described in the section "CDF Estimates with Regression Effects" on page 177.

Parameter Estimation Method

PROC HPSEVERITY uses the maximum likelihood (ML) method to estimate the parameters of each model. A nonlinear optimization process is used to maximize the log of the likelihood function.

Likelihood Function

Let $f_\Theta(x)$ and $F_\Theta(x)$ denote the PDF and CDF, respectively, evaluated at x for a set of parameter values Θ. Let Y denote the random response variable, and let y denote its value recorded in an observation in the input data set. Let T^l and T^r denote the random variables for the left-truncation and right-truncation threshold, respectively, and let t^l and t^r denote their values for an observation, respectively. If there is no left-truncation, then $t^l = \tau^l$, where τ^l is the smallest value in the support of the distribution; so $F(t^l) = 0$. If there is no right-truncation, then $t^r = \tau_h$, where τ_h is the largest value in the support of the distribution; so $F(t^r) = 1$.

Let C^l and C^r denote the random variables for the left-censoring and right-censoring limit, respectively, and let c^l and c^r denote their values for an observation, respectively. If there is no left-censoring, then $c^l = \tau_h$; so $F(c^l) = 1$. If there is no right-censoring, then $c^r = \tau^l$; so $F(c^r) = 0$.

The set of input observations can be categorized into the following four subsets:

- E is the set of uncensored and untruncated observations. The likelihood of an observation in E is

  $$l_E = \Pr(Y = y) = f_\Theta(y)$$

- E_t is the set of uncensored observations that are truncated. The likelihood of an observation in E_t is

  $$l_{E_t} = \Pr(Y = y | t^l < Y \leq t^r) = \frac{f_\Theta(y)}{F_\Theta(t^r) - F_\Theta(t^l)}$$

- C is the set of censored observations that are not truncated. The likelihood of an observation C is

  $$l_C = \Pr(c^r < Y \leq c^l) = F_\Theta(c^l) - F_\Theta(c^r)$$

- C_t is the set of censored observations that are truncated. The likelihood of an observation C_t is

  $$l_{C_t} = \Pr(c^r < Y \leq c^l | t^l < Y \leq t^r) = \frac{F_\Theta(c^l) - F_\Theta(c^r)}{F_\Theta(t^r) - F_\Theta(t^l)}$$

Note that $(E \cup E_t) \cap (C \cup C_t) = \emptyset$. Also, the sets E_t and C_t are empty when no truncation is specified, and the sets C and C_t are empty when no censoring is specified.

Given this, the likelihood of the data L is as follows:

$$L = \prod_E f_\Theta(y) \prod_{E_t} \frac{f_\Theta(y)}{F_\Theta(t^r) - F_\Theta(t^l)} \prod_C F_\Theta(c^l) - F_\Theta(c^r) \prod_{C_t} \frac{F_\Theta(c^l) - F_\Theta(c^r)}{F_\Theta(t^r) - F_\Theta(t^l)}$$

The maximum likelihood procedure used by PROC HPSEVERITY finds an optimal set of parameter values $\hat{\Theta}$ that maximizes $\log(L)$ subject to the boundary constraints on parameter values. For a distribution *dist*, such boundary constraints can be specified by using the *dist*_LOWERBOUNDS and *dist*_UPPERBOUNDS subroutines. For more information, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 192. Some aspects of the optimization process can be controlled by using the NLOPTIONS statement.

Estimating Covariance and Standard Errors

PROC HPSEVERITY computes an estimate of the covariance matrix of the parameters by using the asymptotic theory of the maximum likelihood estimators (MLE). If N denotes the number of observations used for estimating a parameter vector $\boldsymbol{\theta}$, then the theory states that as $N \to \infty$, the distribution of $\hat{\boldsymbol{\theta}}$, the estimate of $\boldsymbol{\theta}$, converges to a normal distribution with mean $\boldsymbol{\theta}$ and covariance $\hat{\mathbf{C}}$ such that $\mathbf{I}(\boldsymbol{\theta}) \cdot \hat{\mathbf{C}} \to 1$, where $\mathbf{I}(\boldsymbol{\theta}) = -E\left[\nabla^2 \log(L(\boldsymbol{\theta}))\right]$ is the information matrix for the likelihood of the data, $L(\boldsymbol{\theta})$. The covariance estimate is obtained by using the inverse of the information matrix.

In particular, if $\mathbf{G} = \nabla^2 \log(-L(\boldsymbol{\theta}))$ denotes the Hessian matrix of the negative of log likelihood, then the covariance estimate is computed as

$$\hat{\mathbf{C}} = \frac{N}{d}\mathbf{G}^{-1}$$

where d is a denominator that is determined by the VARDEF= option. If VARDEF=N, then $d = N$, which yields the asymptotic covariance estimate. If VARDEF=DF, then $d = N - k$, where k is number of parameters (the model's degrees of freedom). The VARDEF=DF option is the default, because it attempts to correct the potential bias introduced by the finite sample.

The standard error s_i of the parameter θ_i is computed as the square root of the ith diagonal element of the estimated covariance matrix; that is, $s_i = \sqrt{\hat{C}_{ii}}$.

If you have specified a custom objective function, then the covariance matrix of the parameters is still computed by inverting the information matrix, except that the Hessian matrix $\mathbf{G}$ is computed as $\mathbf{G} = \nabla^2 \log(U(\boldsymbol{\theta}))$, where U denotes your custom objective function that is minimized by the optimizer.

Covariance and standard error estimates might not be available if the Hessian matrix is found to be singular at the end of the optimization process. This can especially happen if the optimization process stops without converging.

Details of Optimization Algorithms

Overview

There are several optimization techniques available. You can choose a particular optimizer with the TECH=*name* option in the PROC statement or NLOPTIONS statement.

Table 5.4 Optimization Techniques

Algorithm	**TECH=**
Trust region Method	TRUREG
Newton-Raphson method with line search	NEWRAP
Newton-Raphson method with ridging	NRRIDG
Quasi-Newton methods (DBFGS, DDFP, BFGS, DFP)	QUANEW
Double-dogleg method (DBFGS, DDFP)	DBLDOG
Conjugate gradient methods (PB, FR, PR, CD)	CONGRA
Nelder-Mead simplex method	NMSIMP

No algorithm for optimizing general nonlinear functions exists that always finds the global optimum for a general nonlinear minimization problem in a reasonable amount of time. Since no single optimization technique is invariably superior to others, NLO provides a variety of optimization techniques that work well in various circumstances. However, you can devise problems for which none of the techniques in NLO can find the correct solution. Moreover, nonlinear optimization can be computationally expensive in terms of time and memory, so you must be careful when matching an algorithm to a problem.

All optimization techniques in NLO use $O(n^2)$ memory except the conjugate gradient methods, which use only $O(n)$ of memory and are designed to optimize problems with many parameters. These iterative techniques require repeated computation of the following:

- the function value (optimization criterion)
- the gradient vector (first-order partial derivatives)
- for some techniques, the (approximate) Hessian matrix (second-order partial derivatives)

However, since each of the optimizers requires different derivatives, some computational efficiencies can be gained. Table 5.5 shows which derivatives are required for each optimization technique. (FOD means that first-order derivatives or the gradient is computed; SOD means that second-order derivatives or the Hessian is computed.)

Table 5.5 Optimization Computations

Algorithm	**FOD**	**SOD**
TRUREG	x	x
NEWRAP	x	x
NRRIDG	x	x
QUANEW	x	-
DBLDOG	x	-
CONGRA	x	-
NMSIMP	-	-

Each optimization method uses one or more convergence criteria that determine when it has converged. The various termination criteria are listed and described in the previous section. An algorithm is considered to have converged when any one of the convergence criteria is satisfied. For example, under the default settings, the QUANEW algorithm converges if ABSGCONV $< 1\text{E}{-}5$, FCONV $< 10^{-\text{FDIGITS}}$, or GCONV $< 1\text{E}{-}8$.

Choosing an Optimization Algorithm

The factors for choosing a particular optimization technique for a particular problem are complex and might involve trial and error.

For many optimization problems, computing the gradient takes more computer time than computing the function value, and computing the Hessian sometimes takes *much* more computer time and memory than computing the gradient, especially when there are many decision variables. Unfortunately, optimization techniques that do not use some kind of Hessian approximation usually require many more iterations than techniques that do use a Hessian matrix. As a result the total run time of these techniques is often longer. Techniques that do not use the Hessian also tend to be less reliable. For example, they can more easily terminate at stationary points rather than at global optima.

A few general remarks about the various optimization techniques follow:

- The second-derivative methods TRUREG, NEWRAP, and NRRIDG are best for small problems where the Hessian matrix is not expensive to compute. Sometimes the NRRIDG algorithm can be faster than the TRUREG algorithm, but TRUREG can be more stable. The NRRIDG algorithm requires only one matrix with $n(n+1)/2$ double words; TRUREG and NEWRAP require two such matrices.
- The first-derivative methods QUANEW and DBLDOG are best for medium-sized problems where the objective function and the gradient are much faster to evaluate than the Hessian. The QUANEW and

DBLDOG algorithms generally require more iterations than TRUREG, NRRIDG, and NEWRAP, but each iteration can be much faster. The QUANEW and DBLDOG algorithms require only the gradient to update an approximate Hessian, and they require slightly less memory than TRUREG or NEWRAP (essentially one matrix with $n(n+1)/2$ double words). QUANEW is the default optimization method.

- The first-derivative method CONGRA is best for large problems where the objective function and the gradient can be computed much faster than the Hessian and where too much memory is required to store the (approximate) Hessian. The CONGRA algorithm generally requires more iterations than QUANEW or DBLDOG, but each iteration can be much faster. Since CONGRA requires only a factor of n double-word memory, many large applications can be solved only by CONGRA.
- The no-derivative method NMSIMP is best for small problems where derivatives are not continuous or are very difficult to compute.

Parameter Initialization

PROC HPSEVERITY enables you to initialize parameters of a model in different ways. A model can have two kinds of parameters: distribution parameters and regression parameters.

The distribution parameters can be initialized by using one of the following three methods:

INIT= option	You can use the INIT= option in the DIST statement.
INEST= data set	You can use the INEST= data set.
PARMINIT subroutine	You can define a *dist*_PARMINIT subroutine in the distribution model. For more information, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 192.

Note that only one of the initialization methods is used. You cannot combine them. They are used in the following order:

- The method that uses the INIT= option takes the highest precedence. If you use the INIT= option to provide an initial value for at least one parameter, then other initialization methods (INEST= and PARMINIT) are not used. If you specify initial values for some but not all the parameters by using the INIT= option, then the uninitialized parameters are initialized to the default value of 0.001.

  If this option is used when regression effects are specified, then the value of the first distribution parameter must be related to the initial value for the *base* value of the scale or log-transformed scale parameter. For more information, see the section "Estimating Regression Effects" on page 175.
- The method that uses the INEST= data set takes the second precedence. If there is a nonmissing value specified for even one distribution parameter, then the PARMINIT method is not used and any uninitialized parameters are initialized to the default value of 0.001.
- If none of the distribution parameters are initialized by using the INIT= option or the INEST= data set, but the distribution model defines a PARMINIT subroutine, then PROC HPSEVERITY invokes that subroutine with appropriate inputs to initialize the parameters. If the PARMINIT subroutine returns missing values for some parameters, then those parameters are initialized to the default value of 0.001.

- If none of the initialization methods are used, each distribution parameter is initialized to the default value of 0.001.

For more information about regression models and initialization of regression parameters, see the section "Estimating Regression Effects" on page 175.

PARMINIT-Based Parameter Initialization Method and Distributed Data

If you have specified a distributed mode of execution for the procedure, then the input data are distributed across the computational nodes. For more information about the distributed computing model, see the section "Distributed and Multithreaded Computation" on page 189. If the PARMINIT subroutine is used for initializing the distribution parameters, then PROC HPSEVERITY invokes that subroutine on each computational node with the data that are local to that node. The EDF estimates that are supplied to the PARMINIT subroutine are also computed using the local data. The initial values of the parameters that are supplied to the optimizer are the average of the local estimates that are computed on each node. This approach works well if the data are distributed randomly across nodes. If you distribute the data on the appliance before you run the procedure (alongside-the-database model), then you should try to make the distribution as random as possible in order to increase the chances of computing good initial values. If you specify a data set that is not distributed before you run the procedure, then PROC HPSEVERITY distributes the data for you by sending the first observation to the first node, the second observation to the second node, and so on. If the order of observations is random, then this method ensures random distribution of data across the computational nodes.

Estimating Regression Effects

The HPSEVERITY procedure enables you to estimate the effects of regressor (exogenous) variables while fitting a distribution if the distribution has a scale parameter or a log-transformed scale parameter.

Let x_j, $j = 1, \ldots, k$, denote the k regressor variables. Let β_j denote the regression parameter that corresponds to the regressor x_j. If regression effects are not specified, then the model for the response variable Y is of the form

$$Y \sim \mathcal{F}(\Theta)$$

where $\mathcal{F}$ is the distribution of Y with parameters Θ. This model is usually referred to as the error model. The regression effects are modeled by extending the error model to the following form:

$$Y \sim \exp(\sum_{j=1}^{k} \beta_j x_j) \cdot \mathcal{F}(\Theta)$$

Under this model, the distribution of Y is valid and belongs to the same parametric family as $\mathcal{F}$ if and only if $\mathcal{F}$ has a scale parameter. Let θ denote the scale parameter and Ω denote the set of nonscale distribution parameters of $\mathcal{F}$. Then the model can be rewritten as

$$Y \sim \mathcal{F}(\theta, \Omega)$$

such that θ is affected by the regressors as

$$\theta = \theta_0 \cdot \exp(\sum_{j=1}^{k} \beta_j x_j)$$

where θ_0 is the *base* value of the scale parameter. Thus, the regression model consists of the following parameters: θ_0, Ω, and $\beta_j (j = 1, \ldots, k)$.

Given this form of the model, distributions without a scale parameter cannot be considered when regression effects are to be modeled. If a distribution does not have a direct scale parameter, then PROC HPSEVERITY accepts it only if it has a log-transformed scale parameter — that is, if it has a parameter $p = \log(\theta)$.

Parameter Initialization for Regression Models

The regression parameters are initialized either by using the values you specify or by the default method.

- If you provide initial values for the regression parameters, then you must provide valid, nonmissing initial values for θ_0 and β_j parameters for all j.

  You can specify the initial value for θ_0 using either the INEST= data set or the INIT= option in the DIST statement. If the distribution has a direct scale parameter (no transformation), then the initial value for the first parameter of the distribution is used as an initial value for θ_0. If the distribution has a log-transformed scale parameter, then the initial value for the first parameter of the distribution is used as an initial value for $\log(\theta_0)$.

  You can use only the INEST= data set to specify the initial values for β_j. The INEST= data set must contain nonmissing initial values for all the regressors specified in the SCALEMODEL statement. The only missing value allowed is the special missing value .R, which indicates that the regressor is linearly dependent on other regressors. If you specify .R for a regressor for one distribution, you must specify it so for all the distributions.

- If you do not specify valid initial values for θ_0 or β_j parameters for all j, then PROC HPSEVERITY initializes those parameters using the following method:

  Let a random variable Y be distributed as $\mathcal{F}(\theta, \Omega)$, where θ is the scale parameter. By the definition of the scale parameter, a random variable $W = Y/\theta$ is distributed as $\mathcal{G}(\Omega)$ such that $\mathcal{G}(\Omega) = \mathcal{F}(1, \Omega)$. Given a random error term e that is generated from a distribution $\mathcal{G}(\Omega)$, a value y from the distribution of Y can be generated as

  $$y = \theta \cdot e$$

  Taking the logarithm of both sides and using the relationship of θ with the regressors yields:

  $$\log(y) = \log(\theta_0) + \sum_{j=1}^{k} \beta_j x_j + \log(e)$$

  PROC HPSEVERITY makes use of the preceding relationship to initialize parameters of a regression model with distribution *dist* as follows:

  1. The following linear regression problem is solved to obtain initial estimates of β_0 and β_j:

     $$\log(y) = \beta_0 + \sum_{j=1}^{k} \beta_j x_j$$

     The estimates of $\beta_j (j = 1, \ldots, k)$ in the solution of this regression problem are used to initialize the respective regression parameters of the model. The estimate of β_0 is later used to initialize the value of θ_0.

The results of this regression are also used to detect whether any regressors are linearly dependent on the other regressors. If any such regressors are found, then a warning is written to the SAS log and the corresponding regressor is eliminated from further analysis. The estimates for linearly dependent regressors are denoted by a special missing value of .R in the OUTEST= data set and in any displayed output.

2. Let s_0 denote the initial value of the scale parameter.

   If the distribution model of *dist* does not contain the *dist*_PARMINIT subroutine, then s_0 and all the nonscale distribution parameters are initialized to the default value of 0.001.

   However, it is strongly recommended that each distribution's model contain the *dist*_PARMINIT subroutine. For more information, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 192. If that subroutine is defined, then s_0 is initialized as follows: Each input value y_i of the response variable is transformed to its scale-normalized version w_i as

$$w_i = \frac{y_i}{\exp(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij})}$$

   where x_{ij} denotes the value of jth regressor in the ith input observation. These w_i values are used to compute the input arguments for the *dist*_PARMINIT subroutine. The values that are computed by the subroutine for nonscale parameters are used as their respective initial values. If the distribution has an untransformed scale parameter, then s_0 is set to the value of the scale parameter that is computed by the subroutine. If the distribution has a log-transformed scale parameter P, then s_0 is computed as $s_0 = \exp(l_0)$, where l_0 is the value of P computed by the subroutine.

3. The value of θ_0 is initialized as

$$\theta_0 = s_0 \cdot \exp(\beta_0)$$

Reporting Estimates of Regression Parameters

When you request estimates to be written to the output (either ODS displayed output or in the OUTEST= data set), the estimate of the base value of the first distribution parameter is reported. If the first parameter is the log-transformed scale parameter, then the estimate of $\log(\theta_0)$ is reported; otherwise, the estimate of θ_0 is reported. The transform of the first parameter of a distribution *dist* is controlled by the *dist*_SCALETRANSFORM function that is defined for it.

CDF Estimates with Regression Effects

When regression effects are estimated, the estimate of the scale parameter depends on the values of the regressors and the estimates of the regression parameters. This dependency results in a potentially different distribution for each observation. PROC HPSEVERITY needs to compute the estimates of the cumulative distribution function (CDF) to compute the EDF-based statistics of fit that compare the nonparametric and parametric estimates of the distribution function. To make the CDF estimates comparable across distributions and comparable to the empirical distribution function (EDF), PROC HPSEVERITY computes the CDF estimates from a representative distribution. The *representative distribution* is a mixture of a certain number of distributions, where each distribution differs only in the value of the scale parameter. You can specify the number of distributions in the mixture and how their scale values are chosen by using the DFMIXTURE= option in the SCALEMODEL statement.

Let N denote the number of observations used for EDF estimation, K denote the number of components in the mixture distribution, s_k denote the scale parameter of the kth mixture component, and d_k denote the weight associated with kth mixture component.

Let $F(y; s_k, \hat{\Omega})$ denote the CDF of the kth component distribution, where $\hat{\Omega}$ denotes the set of estimates of all parameters of the distribution other than the scale parameter. Then, the CDF estimates, $F^*(y)$, of the mixture distribution at y are computed as

$$F^*(y) = \frac{1}{D} \sum_{k=1}^{K} d_k F(y; s_k, \hat{\Omega})$$

where D is the normalization factor ($D = \sum_{k=1}^{K} d_k$). The $F^*(y)$ values are used for computing the EDF-based statistics of fit.

The scale values s_k for the K mixture components are derived from the set $\{\hat{\theta}_i\}$ ($i = 1 \dots N$) of N scale values, where $\hat{\theta}_i$ denotes the estimate of the scale parameter due to observation i. It is computed as

$$\hat{\theta}_i = \hat{\theta}_0 \cdot \exp(\sum_{j=1}^{k} \hat{\beta}_j x_{ij})$$

where $\hat{\theta}_0$ is an estimate of the base value of the scale parameter, $\hat{\beta}_j$ are the estimates of regression coefficients, and x_{ij} is the value of regressor j in observation i.

Let w_i denote the weight of observation i. If the WEIGHT statement is specified, then it is equal to the value of the specified weight variable for the corresponding observation in the DATA= data set; otherwise, it is set to 1.

You can specify one of the following *method-names* in the DFMIXTURE= option in the SCALEMODEL statement to specify the method of choosing K and the corresponding s_k and d_k values:

FULL
: In this method, there are as many mixture components as the number of observations that are used for estimation. In other words, $K = N$, $s_k = \hat{\theta}_k$, and $d_k = w_k$ ($k = 1 \dots N$). This is the slowest method, because it requires $O(N)$ computations to compute the mixture CDF $F^*(y_i)$ of one observation. For N observations, the computational complexity in terms of number of CDF evaluations is $O(N^2)$. Even for moderately large values of N, the time taken to compute the mixture CDF can significantly exceed the time taken to estimate the model parameters. So, it is recommended that you use the FULL method only for small data sets.

MEAN
: In this method, the mixture contains only one distribution, whose scale value is the mean of the scale values that are implied by all the observations. In other words, s_1 is computed as

$$s_1 = \frac{1}{W} \sum_{i=1}^{N} w_i \hat{\theta}_i$$

where W is the total weight ($W = \sum_{i=1}^{N} w_i$).

This method is the fastest because it requires only one CDF evaluation per observation. The computational complexity is $O(N)$ for N observations.

If you do not specify the DFMIXTURE= option in the SCALEMODEL statement, then this is the default method.

QUANTILE	In this method, a certain number of quantiles are chosen from the set of all scale values. If you specify a value of q for the K= option when specifying this method, then $K = q - 1$ and s_k are set to be the $(q - 1)$ q-quantiles from the set $\{\hat{\theta}_i\}$ $(i = 1 \ldots N)$. The weight of each of the components (d_k) is assumed to be 1 for this method. The default value of q is 2, which implies a one-point mixture that has a distribution whose scale value is equal to the median scale value. For this method, PROC HPSEVERITY needs to sort the N scale values in the set $\{\hat{\theta}_i\}$; the sorting requires $O(N \log(N))$ computations. Then, computing the mixture estimate of one observation requires $(q-1)$ CDF evaluations. Hence, the computational complexity of this method is $O(qN) + O(N \log(N))$ for computing a mixture CDF of N observations. For $q << N$, the QUANTILE method is significantly faster than the FULL method.
RANDOM	In this method, a uniform random sample of observations is chosen, and the mixture contains the distributions that are implied by those observations. If you specify a value of r for the K= option when specifying this method, then the size of the sample is r. Hence, $K = r$. If l_j denotes the index of jth observation in the sample $(j = 1 \ldots r)$, such that $1 \leq l_j \leq N$, then the scale of kth component distribution in the mixture is $s_k = \hat{\theta}_{l_k}$ and the weight associated with it is $d_k = w_{l_k}$. You can also specify the seed to be used for generating the random sample by using the SEED= option for this method. The same sample of observations is used for all models. Computing a mixture estimate of one observation requires r CDF evaluations. Hence, the computational complexity of this method is $O(rN)$ for computing a mixture CDF of N observations. For $r << N$, the RANDOM method is significantly faster than the FULL method.

Empirical Distribution Function Estimation Methods

The empirical distribution function (EDF) is a nonparametric estimate of the cumulative distribution function (CDF) of the distribution. PROC HPSEVERITY computes EDF estimates for two purposes: to send the estimates to a distribution's PARMINIT subroutine in order to initialize the distribution parameters, and to compute the EDF-based statistics of fit.

To reduce the time that it takes to compute the EDF estimates, you can use the INITSAMPLE option to specify that only a fraction of the input data be used. If you do not specify the INITSAMPLE option and the data set has more than 10,000 valid observations, then a uniform random sample of at most 10,000 observations is used for EDF estimation.

In the distributed mode of execution, in which data are distributed across the grid nodes, the EDF estimates are computed on each node by using the portion of the input data that is located on that node. These local EDF estimates are an approximation of the global EDF estimates, which would been computed by using the entire input data set. PROC HPSEVERITY does not compute global EDF estimates. Let X denote a quantity that depends on the EDF estimates. X can be either an EDF-based initial value of a distribution parameter or an EDF-based statistic of fit. PROC HPSEVERITY estimates X as follows: First, each grid node k computes an estimate X_k by using the local EDF estimates that are computed on that node. Then, the estimate $\hat{X}$ of X is computed as an average of all the X_k values; that is, $\hat{X} = \sum_{i=1}^{K} X_k$, where K denotes the total number of nodes where the data reside.

This section describes the methods that are used for computing EDF estimates.

Notation

Let there be a set of N observations, each containing a quintuplet of values $(y_i, t_i^l, t_i^r, c_i^r, c_i^l), i = 1, \ldots, N$, where y_i is the value of the response variable, t_i^l is the value of the left-truncation threshold, t_i^r is the value of the right-truncation threshold, and c_i^r is the value of the right-censoring limit, c_i^l is the value of the left-censoring limit.

If an observation is not left-truncated, then $t_i^l = \tau^l$, where τ^l is the smallest value in the support of the distribution; so $F(t_i^l) = 0$. If an observation is not right-truncated, then $t_i^r = \tau_h$, where τ_h is the largest value in the support of the distribution; so $F(t_i^r) = 1$. If an observation is not right-censored, then $c_i^r = \tau^l$; so $F(c_i^r) = 0$. If an observation is not left-censored, then $c_i^l = \tau_h$; so $F(c_i^l) = 1$.

Let w_i denote the weight associated with ith observation. If you have specified the WEIGHT statement, then w_i is the normalized value of the weight variable; otherwise, it is set to 1. The weights are normalized such that they sum up to N.

An indicator function $I[e]$ takes a value of 1 or 0 if the expression e is true or false, respectively.

Estimation Methods

If the response variable is subject to both left-censoring and right-censoring effects and if you have not specified the EMPIRICALCDF=NOTURNBULL method, then PROC HPSEVERITY uses the Turnbull's method. This section describes methods other than Turnbull's method. For Turnbull's method, see the next section "Turnbull's EDF Estimation Method (Experimental)" on page 182.

The method descriptions assume that all observations are either uncensored or right-censored; that is, each observation is of the form $(y_i, t_i^l, t_i^r, \tau^l, \tau_h)$ or $(y_i, t_i^l, t_i^r, c_i^r, \tau_h)$.

If all observations are either uncensored or left-censored, then each observation is of the form $(y_i, t_i^l, t_i^r, \tau_l, c_i^l)$. It is converted to an observation $(-y_i, -t_i^r, -t_i^l, -c_i^l, \tau_h)$; that is, the signs of all the response variable values are reversed, the new left-truncation threshold is equal to the negative of the original right-truncation threshold, the new right-truncation threshold is equal to the negative of the original left-truncation threshold, and the negative of the original left-censoring limit becomes the new right-censoring limit. With this transformation, each observation is either uncensored or right-censored. The methods described for handling uncensored or right-censored data are now applicable. After the EDF estimates are computed, the observations are transformed back to the original form and EDF estimates are adjusted such $F_n(y_i) = 1 - F_n(-y_i-)$, where $F_n(-y_i-)$ denotes the EDF estimate of the value slightly less than the transformed value $-y_i$.

Further, a set of uncensored or right-censored observations can be converted to a set of observations of the form $(y_i, t_i^l, t_i^r, \delta_i)$, where δ_i is the indicator of right-censoring. $\delta_i = 0$ indicates a right-censored observation, in which case y_i is assumed to record the right-censoring limit c_i^r. $\delta_i = 1$ indicates an uncensored observation, and y_i records the exact observed value. In other words, $\delta_i = I[Y \leq C^r]$ and $y_i = \min(y_i, c_i^r)$.

Given this notation, the EDF is estimated as

$$F_n(y) = \begin{cases} 0 & \text{if } y < y^{(1)} \\ \hat{F}_n(y^{(k)}) & \text{if } y^{(k)} \leq y < y^{(k+1)}, k = 1, \ldots, N-1 \\ \hat{F}_n(y^{(N)}) & \text{if } y^{(N)} \leq y \end{cases}$$

where $y^{(k)}$ denotes the kth order statistic of the set $\{y_i\}$ and $\hat{F}_n(y^{(k)})$ is the estimate computed at that value. The definition of $\hat{F}_n$ depends on the estimation method. You can specify a particular method or let

PROC HPSEVERITY choose an appropriate method by using the EMPIRICALCDF= option in the PROC HPSEVERITY statement. Each method computes $\hat{F}_n$ as follows:

NOTURNBULL This is the default method. First, censored observations, if any, are processed as follows:

- An observation that is left-censored but not right-censored is converted to an uncensored observation $(y_i^u, t_i^l, t_i^r, \tau^l, \tau_h)$, where $y_i^u = c_i^l/2$.
- An observation that is both left-censored and right-censored is converted to an uncensored observation $(y_i^u, t_i^l, t_i^r, \tau^l, \tau_h)$, where $y_i^u = (c_i^r + c_i^l)/2$.
- An observation that is right-censored but not left-censored is left unchanged.

If the processed set of observations contains any truncated or right-censored observations, the KAPLANMEIER method is used. Otherwise, the STANDARD method is used.

The observations are modified only for the purpose of computing the EDF estimates. The original censoring information is used by the parameter estimation process.

STANDARD This method is chosen when no censoring or truncation information is specified. It is the standard way of computing EDF. The EDF estimate at observation i is computed as follows:

$$\hat{F}_n(y_i) = \frac{1}{N}\sum_{j=1}^{N} I[y_j \leq y_i]$$

KAPLANMEIER This method is chosen when at least one form of censoring or truncation is specified. The Kaplan-Meier (KM) estimator, also known as the product-limit estimator, was first introduced by Kaplan and Meier (1958) for censored data. Lynden-Bell (1971) derived a similar estimator for left-truncated data. PROC HPSEVERITY uses the definition that combines both censoring and truncation information (Klein and Moeschberger 1997; Lai and Ying 1991).

The EDF estimate at observation i is computed as

$$\hat{F}_n(y_i) = 1 - \prod_{\tau \leq y_i}\left(1 - \frac{n_\tau}{R_n(\tau)}\right)$$

where n_τ and $R_n(\tau)$ are defined as follows:

- $n_\tau = \sum_{k=1}^{N} w_k \cdot I[y_k = \tau \text{ and } \tau \leq t_k^r \text{ and } \delta_k = 1]$, which is the number of uncensored observations ($\delta_k = 1$) for which the response variable value is equal to τ and τ is observable according to the right-truncation threshold of that observation ($\tau \leq t_k^r$).
- $R_n(\tau) = \sum_{k=1}^{N} w_k \cdot I[y_k \geq \tau > t_k^l]$, which is the size (cardinality) of the risk set at τ. The term *risk set* has its origins in survival analysis; it contains the events that are at risk of failure at a given time, τ. In other words, it contains the events that have survived up to time τ and might fail at or after τ. For PROC HPSEVERITY, *time* is equivalent to the magnitude of the event and *failure* is equivalent to an uncensored and observable event, where observable means it satisfies the truncation thresholds.

MODIFIEDKM

The product-limit estimator used by the KAPLANMEIER method does not work well if the risk set size becomes very small. For right-censored data, the size can become small towards the right tail. For left-truncated data, the size can become small at the left tail and can remain so for the entire range of data. This was demonstrated by Lai and Ying (1991). They proposed a modification to the estimator that ignores the effects due to small risk set sizes.

The EDF estimate at observation i is computed as

$$\hat{F}_n(y_i) = 1 - \prod_{\tau \le y_i} \left(1 - \frac{n(\tau)}{R_n(\tau)} \cdot I[R_n(\tau) \ge cN^{\alpha}] \right)$$

where the definitions of $n(\tau)$ and $R_n(\tau)$ are identical to those used for the KAPLANMEIER method described previously.

You can specify the values of c and α by using the C= and ALPHA= options. If you do not specify a value for c, the default value used is $c = 1$. If you do not specify a value for α, the default value used is $\alpha = 0.5$.

As an alternative, you can also specify an absolute lower bound, say L, on the risk set size by using the RSLB= option, in which case $I[R_n(\tau) \ge cN^{\alpha}]$ is replaced by $I[R_n(\tau) \ge L]$ in the definition.

Turnbull's EDF Estimation Method (Experimental)

If the response variable is subject to both left-censoring and right-censoring effects and if you have not specified the NOTURNBULL method, then the HPSEVERITY procedure uses a method proposed by Turnbull (1976) to compute the nonparametric estimates of the cumulative distribution function. The original Turnbull's method is modified using the suggestions made by Frydman (1994) when truncation effects are present.

Let the input data consist of N observations in the form of quintuplets of values $(y_i, t_i^l, t_i^r, c_i^r, c_i^l), i = 1, \ldots, N$ with notation described in the section "Notation" on page 180. For each observation, let $A_i = (c_i^r, c_i^l]$ be the censoring interval; that is, the response variable value is known to lie in the interval A_i, but the exact value is not known. If an observation is uncensored, then $A_i = (y_i - \epsilon, y_i]$ for any arbitrarily small value of $\epsilon > 0$. If an observation is censored, then the value y_i is ignored. Similarly, for each observation, let $B_i = (t_i^l, t_i^r]$ be the truncation interval; that is, the observation is drawn from a truncated (conditional) distribution $F(y, B_i) = P(Y \le y | Y \in B_i)$.

Two sets, L and R, are formed using A_i and B_i as follows:

$$L = \{c_i^r, 1 \le i \le N\} \cup \{t_i^r, 1 \le i \le N\}$$
$$R = \{c_i^l, 1 \le i \le N\} \cup \{t_i^l, 1 \le i \le N\}$$

The sets L and R represent the left endpoints and right endpoints, respectively. A set of disjoint intervals $C_j = [q_j, p_j], 1 \le j \le M$ is formed such that $q_j \in L$ and $p_j \in R$ and $q_j \le p_j$ and $p_j < q_{j+1}$. The value of M is dependent on the nature of censoring and truncation intervals in the input data. Turnbull (1976) showed that the maximum likelihood estimate (MLE) of the EDF can increase only inside intervals C_j. In other words, the MLE estimate is constant in the interval (p_j, q_{j+1}). The likelihood is independent of the behavior of F_n inside any of the intervals C_j. Let s_j denote the increase in F_n inside an interval C_j. Then, the EDF estimate is as follows:

$$F_n(y) = \begin{cases} 0 & \text{if } y < q_1 \\ \sum_{k=1}^{j} s_k & \text{if } p_j < y < q_{j+1}, 1 \le j \le M-1 \\ 1 & \text{if } y > p_M \end{cases}$$

PROC HPSEVERITY computes the estimates $F_n(p_j+) = F_n(q_{j+1}-) = \sum_{k=1}^{j} s_k$ at points p_j and q_{j+1} and computes $F_n(q_1-) = 0$ at point q_1, where $F_n(x+)$ denotes the limiting estimate at a point that is infinitesimally larger than x when approaching x from values larger than x and where $F_n(x-)$ denotes the limiting estimate at a point that is infinitesimally smaller than x when approaching x from values smaller than x.

PROC HPSEVERITY uses the expectation-maximization (EM) algorithm proposed by Turnbull (1976), who referred to the algorithm as the self-consistency algorithm. By default, the algorithm runs until one of the following criteria is met:

- Relative-error criterion: The maximum relative error between the two consecutive estimates of s_j falls below a threshold ϵ. If l indicates an index of the current iteration, then this can be formally stated as

  $$\arg \max_{1 \le j \le M} \left\{ \frac{|s_j^l - s_j^{l-1}|}{s_j^{l-1}} \right\} \le \epsilon$$

  You can control the value of ϵ by specifying the EPS= suboption of the EDF=TURNBULL option in the PROC HPSEVERITY statement. The default value is 1.0E–8.

- Maximum-iteration criterion: The number of iterations exceeds an upper limit specified by the MAXITER= suboption of the EDF=TURNBULL option in the PROC HPSEVERITY statement. The default number of maximum iterations is 500.

The self-consistent estimates obtained in this manner might not be maximum likelihood estimates. Gentleman and Geyer (1994) suggested the use of the Kuhn-Tucker conditions for the maximum likelihood problem to ensure that the estimates are MLE. If you specify the ENSUREMLE suboption of the EDF=TURNBULL option in the PROC HPSEVERITY statement, then PROC HPSEVERITY computes the Kuhn-Tucker conditions at the end of each iteration to determine whether the estimates $\{s_j\}$ are MLE. If no truncation effects are specified, then the Kuhn-Tucker conditions derived by Gentleman and Geyer (1994) are used. If truncation effects are specified, then PROC HPSEVERITY uses modified Kuhn-Tucker conditions that account for the truncation effects. An integral part of checking the conditions is to determine whether an estimate s_j is zero or whether an estimate of the Lagrange multiplier or the reduced gradient associated with the estimate s_j is zero. PROC HPSEVERITY declares these values to be zero if they are less than or equal to a threshold δ. You can control the value of δ by specifying the ZEROPROB= suboption of the EDF=TURNBULL option in the PROC HPSEVERITY statement. The default value is 1.0E–8. The algorithm continues until the Kuhn-Tucker conditions are satisfied or the number of iterations exceeds the upper limit. The relative-error criterion stated previously is not used when the ENSUREMLE option is specified.

EDF Estimates and Truncation

If truncation is specified, then the estimate $\hat{F}_n(y)$ computed by any method other than the STANDARD method is a *conditional* estimate. In other words, $\hat{F}_n(y) = \Pr(Y \leq y | \tau_G < Y \leq \tau_H)$, where G and H denote the (unknown) distribution functions of the left-truncation threshold variable T^l and the right-truncation threshold variable T^r, respectively, τ_G denotes the smallest left-truncation threshold with a nonzero cumulative probability, and τ_H denotes the largest right-truncation threshold with a nonzero cumulative probability. Formally, $\tau_G = \inf\{s : G(s) > 0\}$ and $\tau_H = \sup\{s : H(s) > 0\}$. For computational purposes, PROC HPSEVERITY estimates τ_G and τ_H by t^l_{min} and t^r_{max}, respectively, defined as

$$t^l_{\text{min}} = \min\{t^l_k : 1 \leq k \leq N\}$$
$$t^r_{\text{max}} = \max\{t^r_k : 1 \leq k \leq N\}$$

These estimates of t^l_{min} and t^r_{max} are used to compute the conditional estimates of the CDF as described in the section "Truncation and Conditional CDF Estimates" on page 170.

Supplying EDF Estimates to Functions and Subroutines

The parameter initialization subroutines in distribution models and some predefined utility functions require EDF estimates. For more information, see the sections "Defining a Severity Distribution Model with the FCMP Procedure" on page 192 and "Predefined Utility Functions" on page 204.

PROC HPSEVERITY supplies the EDF estimates to these subroutines and functions by using two arrays, x and F, the dimension of each array, and a type of the EDF estimates. The type identifies how the EDF estimates are computed and stored. They are as follows:

Type 1	specifies that EDF estimates are computed using the STANDARD method; that is, the data used for estimation are neither censored nor truncated.
Type 2	specifies that EDF estimates are computed using the KAPLANMEIER method; that is, the data used for estimation are subject to at least one form of truncation or censoring.
Type 3	specifies that EDF estimates are computed using the TURNBULL method; that is, the data used for estimation are subject to both left- and right-censoring. The data might or might not be truncated.

For Types 1 and 2, the EDF estimates are stored in arrays x and F of dimension N such that the following holds:

$$F_n(y) = \begin{cases} 0 & \text{if } y < x[1] \\ F[k] & \text{if } x[k] \leq y < x[k+1], k = 1, \ldots, N-1 \\ F[N] & \text{if } x[N] \leq y \end{cases}$$

where $[k]$ denotes kth element of the array ([1] denotes the first element of the array).

For Type 3, the EDF estimates are stored in arrays x and F of dimension N such that the following holds:

$$F_n(y) = \begin{cases} 0 & \text{if } y < x[1] \\ \text{undefined} & \text{if } x[2k-1] \leq y < x[2k], k = 1, \ldots, (N-1)/2 \\ F[2k] = F[2k+1] & \text{if } x[2k] \leq y < x[2k+1], k = 1, \ldots, (N-1)/2 \\ F[N] & \text{if } x[N] \leq y \end{cases}$$

Although the behavior of EDF is theoretically undefined for the interval $[x[2k-1], x[2k])$, for computational purposes, all predefined functions and subroutines assume that the EDF increases linearly from $F[2k-1]$

to $F[2k]$ in that interval if $x[2k-1] < x[2k]$. If $x[2k-1] = x[2k]$, which can happen when the EDF is estimated from a combination of uncensored and interval-censored data, the predefined functions and subroutines assume that $F_n(x[2k-1]) = F_n(x[2k]) = F[2k]$.

Statistics of Fit

PROC HPSEVERITY computes and reports various statistics of fit to indicate how well the estimated model fits the data. The statistics belong to two categories: likelihood-based statistics and EDF-based statistics. Neg2LogLike, AIC, AICC, and BIC are likelihood-based statistics, and KS, AD, and CvM are EDF-based statistics.

In the distributed mode of execution, in which data are distributed across the grid nodes, the EDF estimates are computed by using the local data. The EDF-based statistics are computed by using these local EDF estimates. The reported value of each EDF-based statistic is an average of the values of the statistic that are computed by all the grid nodes where the data reside. Also, for large data sets, in both single-machine and distributed modes of execution, the EDF estimates are computed by using a fraction of the input data that is governed by either the INITSAMPLE option or the default sample size. Because of this nature of computing the EDF estimates, the EDF-based statistics of fit are an approximation of the values that would have been computed if the entire input data set were used for computing the EDF estimates. So the values that are reported for EDF-based statistics should be used only for comparing different models. The reported values should not be interpreted as true estimates of the corresponding statistics.

The likelihood-based statistics are reported for the entire input data in both single-machine and distributed modes of execution.

The following subsections provide definitions of each category of statistics.

Likelihood-Based Statistics of Fit

Let $y_i, i = 1, \ldots, N$, denote the response variable values. Let L be the likelihood as defined in the section "Likelihood Function" on page 170. Let p denote the number of model parameters that are estimated. Note that $p = p_d + (k - k_r)$, where p_d is the number of distribution parameters, k is the number of regressors specified in the SCALEMODEL statement, and k_r is the number of regressors found to be linearly dependent (redundant) on other regressors. Given this notation, the likelihood-based statistics are defined as follows:

Neg2LogLike The log likelihood is reported as

$$\text{Neg2LogLike} = -2\log(L)$$

The multiplying factor -2 makes it easy to compare it to the other likelihood-based statistics. A model that has a smaller value of Neg2LogLike is deemed better.

AIC Akaike's information criterion (AIC) is defined as

$$\text{AIC} = -2\log(L) + 2p$$

A model that has a smaller AIC value is deemed better.

AICC The corrected Akaike's information criterion (AICC) is defined as

$$\text{AICC} = -2\log(L) + \frac{2Np}{N-p-1}$$

A model that has a smaller AICC value is deemed better. It corrects the finite-sample bias that AIC has when N is small compared to p. AICC is related to AIC as

$$\text{AICC} = \text{AIC} + \frac{2p(p+1)}{N-p-1}$$

As N becomes large compared to p, AICC converges to AIC. AICC is usually recommended over AIC as a model selection criterion.

BIC The Schwarz Bayesian information criterion (BIC) is defined as

$$\text{BIC} = -2\log(L) + p\log(N)$$

A model that has a smaller BIC value is deemed better.

EDF-Based Statistics

This class of statistics is based on the difference between the estimate of the cumulative distribution function (CDF) and the estimate of the empirical distribution function (EDF). A model that has a smaller value of the chosen EDF-based statistic is deemed better.

Let $y_i, i = 1, \ldots, N$ denote the sample of N values of the response variable. Let $r_i = \sum_{j=1}^{N} I[y_j \leq y_i]$ denote the number of observations with a value less than or equal to y_i, where I is an indicator function. Let $F_n(y_i)$ denote the EDF estimate that is computed by using the method specified in the EMPIRICALCDF= option. Let $Z_i = \hat{F}(y_i)$ denote the estimate of the CDF. Let $F_n(Z_i)$ denote the EDF estimate of Z_i values that are computed using the same method that is used to compute the EDF of y_i values. Using the probability integral transformation, if $F(y)$ is the true distribution of the random variable Y, then the random variable $Z = F(y)$ is uniformly distributed between 0 and 1 (D'Agostino and Stephens 1986, Ch. 4). Thus, comparing $F_n(y_i)$ with $\hat{F}(y_i)$ is equivalent to comparing $F_n(Z_i)$ with $\hat{F}(Z_i) = Z_i$ (uniform distribution).

Note the following two points regarding which CDF estimates are used for computing the test statistics:

- If regressor variables are specified, then the CDF estimates Z_i used for computing the EDF test statistics are from a mixture distribution. See the section "CDF Estimates with Regression Effects" on page 177 for more information.
- If the EDF estimates are conditional because of the truncation information, then each unconditional estimate Z_i is converted to a conditional estimate using the method described in the section "Truncation and Conditional CDF Estimates" on page 170.

In the following, it is assumed that Z_i denotes an appropriate estimate of the CDF if truncation or regression effects are specified. Given this, the EDF-based statistics of fit are defined as follows:

KS The Kolmogorov-Smirnov (KS) statistic computes the largest vertical distance between the CDF and the EDF. It is formally defined as follows:

$$\text{KS} = \sup_y |F_n(y) - F(y)|$$

If the STANDARD method is used to compute the EDF, then the following formula is used:

$$D^{+} = \max_i\left(\frac{r_i}{N} - Z_i\right)$$
$$D^{-} = \max_i\left(Z_i - \frac{r_{i-1}}{N}\right)$$
$$\text{KS} = \sqrt{N}\,\max(D^{+}, D^{-}) + \frac{0.19}{\sqrt{N}}$$

Note that r_0 is assumed to be 0.

If the method used to compute the EDF is any method other than the STANDARD method, then the following formula is used:

$$D^{+} = \max_i(F_n(Z_i) - Z_i),\ \text{if } F_n(Z_i) \geq Z_i$$
$$D^{-} = \max_i(Z_i - F_n(Z_i)),\ \text{if } F_n(Z_i) < Z_i$$
$$\text{KS} = \sqrt{N}\,\max(D^{+}, D^{-}) + \frac{0.19}{\sqrt{N}}$$

AD The Anderson-Darling (AD) statistic is a quadratic EDF statistic that is proportional to the expected value of the weighted squared difference between the EDF and CDF. It is formally defined as follows:

$$\text{AD} = N \int_{-\infty}^{\infty} \frac{(F_n(y) - F(y))^2}{F(y)(1 - F(y))} dF(y)$$

If the STANDARD method is used to compute the EDF, then the following formula is used:

$$\text{AD} = -N - \frac{1}{N}\sum_{i=1}^{N}\left[(2r_i - 1)\log(Z_i) + (2N + 1 - 2r_i)\log(1 - Z_i)\right]$$

If the method used to compute the EDF is any method other than the STANDARD method, then the statistic can be computed by using the following two pieces of information:

- If the EDF estimates are computed using the KAPLANMEIER or MODIFIEDKM methods, then EDF is a step function such that the estimate $F_n(z)$ is a constant equal to $F_n(Z_{i-1})$ in interval $[Z_{i-1}, Z_i]$. If the EDF estimates are computed using the TURNBULL method, then there are two types of intervals: one in which the EDF curve is constant and the other in which the EDF curve is theoretically undefined. For computational purposes, it is assumed that the EDF curve is linear for the latter type of the interval. For each method, the EDF estimate $F_n(y)$ at y can be written as

  $$F_n(z) = F_n(Z_{i-1}) + S_i(z - Z_{i-1}),\ \text{for } z \in [Z_{i-1}, Z_i]$$

  where S_i is the slope of the line defined as

  $$S_i = \frac{F_n(Z_i) - F_n(Z_{i-1})}{Z_i - Z_{i-1}}$$

  For the KAPLANMEIER or MODIFIEDKM method, $S_i = 0$ in each interval.

- Using the probability integral transform $z = F(y)$, the formula simplifies to

$$\text{AD} = N \int_{-\infty}^{\infty} \frac{(F_n(z) - z)^2}{z(1-z)} dz$$

The computation formula can then be derived from the following approximation:

$$\begin{aligned}\text{AD} &= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} \frac{(F_n(z) - z)^2}{z(1-z)} dz \\ &= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} \frac{(F_n(Z_{i-1}) + S_i(z - Z_{i-1}) - z)^2}{z(1-z)} dz \\ &= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} \frac{(P_i - Q_i z)^2}{z(1-z)} dz\end{aligned}$$

where $P_i = F_n(Z_{i-1}) - S_i Z_{i-1}$, $Q_i = 1 - S_i$, and K is the number of points at which the EDF estimate are computed. For the TURNBULL method, $K = 2k$ for some k.

Assuming $Z_0 = 0$, $Z_{K+1} = 1$, $F_n(0) = 0$, and $F_n(Z_K) = 1$ yields the following computation formula:

$$\begin{aligned}\text{AD} = &- N(Z_1 + \log(1 - Z_1) + \log(Z_K) + (1 - Z_K)) \\ &+ N \sum_{i=2}^{K} \left[P_i^2 A_i - (Q_i - P_i)^2 B_i - Q_i^2 C_i\right]\end{aligned}$$

where $A_i = \log(Z_i) - \log(Z_{i-1})$, $B_i = \log(1 - Z_i) - \log(1 - Z_{i-1})$, and $C_i = Z_i - Z_{i-1}$.

If EDF estimates are computed using the KAPLANMEIER or MODIFIEDKM method, then $P_i = F_n(Z_{i-1})$ and $Q_i = 1$, which simplifies the formula as

$$\begin{aligned}\text{AD} = &- N(1 + \log(1 - Z_1) + \log(Z_K)) \\ &+ N \sum_{i=2}^{K} \left[F_n(Z_{i-1})^2 A_i - (1 - F_n(Z_{i-1}))^2 B_i\right]\end{aligned}$$

CvM The Cramér-von Mises (CvM) statistic is a quadratic EDF statistic that is proportional to the expected value of the squared difference between the EDF and CDF. It is formally defined as follows:

$$\text{CvM} = N \int_{-\infty}^{\infty} (F_n(y) - F(y))^2 dF(y)$$

If the STANDARD method is used to compute the EDF, then the following formula is used:

$$\text{CvM} = \frac{1}{12N} + \sum_{i=1}^{N} \left(Z_i - \frac{(2r_i - 1)}{2N} \right)^2$$

If the method used to compute the EDF is any method other than the STANDARD method, then the statistic can be computed by using the following two pieces of information:

- As described previously for the AD statistic, the EDF estimates are assumed to be piecewise linear such that the estimate $F_n(y)$ at y is

$$F_n(z) = F_n(Z_{i-1}) + S_i(z - Z_{i-1}), \text{ for } z \in [Z_{i-1}, Z_i]$$

  where S_i is the slope of the line defined as

$$S_i = \frac{F_n(Z_i) - F_n(Z_{i-1})}{Z_i - Z_{i-1}}$$

  For the KAPLANMEIER or MODIFIEDKM method, $S_i = 0$ in each interval.

- Using the probability integral transform $z = F(y)$, the formula simplifies to:

$$\text{CvM} = N \int_{-\infty}^{\infty} (F_n(z) - z)^2 dz$$

The computation formula can then be derived from the following approximation:

$$\begin{aligned}\text{CvM} &= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} (F_n(z) - z)^2 dz \\ &= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} (F_n(Z_{i-1}) + S_i(z - Z_{i-1}) - z)^2 dz \\ &= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} (P_i - Q_i z)^2 dz\end{aligned}$$

where $P_i = F_n(Z_{i-1}) - S_i Z_{i-1}$, $Q_i = 1 - S_i$, and K is the number of points at which the EDF estimate are computed. For the TURNBULL method, $K = 2k$ for some k.

Assuming $Z_0 = 0$, $Z_{K+1} = 1$, and $F_n(0) = 0$ yields the following computation formula:

$$\text{CvM} = N\frac{Z_1^3}{3} + N \sum_{i=2}^{K+1} \left[P_i^2 A_i - P_i Q_i B_i - \frac{Q_i^2}{3} C_i \right]$$

where $A_i = Z_i - Z_{i-1}$, $B_i = Z_i^2 - Z_{i-1}^2$, and $C_i = Z_i^3 - Z_{i-1}^3$.

If EDF estimates are computed using the KAPLANMEIER or MODIFIEDKM method, then $P_i = F_n(Z_{i-1})$ and $Q_i = 1$, which simplifies the formula as

$$\text{CvM} = \frac{N}{3} + N \sum_{i=2}^{K+1} \left[F_n(Z_{i-1})^2 (Z_i - Z_{i-1}) - F_n(Z_{i-1})(Z_i^2 - Z_{i-1}^2) \right]$$

which is similar to the formula proposed by Koziol and Green (1976).

Distributed and Multithreaded Computation

PROC HPSEVERITY makes an attempt to use all the computational resources that you specify in the PERFORMANCE statement in order to complete the assigned tasks as fast as possible. This section describes the distributed and multithreading computing methods that PROC HPSEVERITY uses.

Distributed Computing

Distributed computing refers to the organization of computation work into multiple tasks that are processed on different nodes; a node is one of the machines that constitute the grid. The number of nodes that PROC HPSEVERITY uses is determined by the distributed processing execution mode. If you specify the client-data (or local-data) mode of execution, then the number of nodes is determined by the NODES= option in the PERFORMANCE statement. If you are using the alongside-the-database mode of execution, then PROC HPSEVERITY determines the number of nodes internally by using the information that is associated with the DATA= data set and the grid information that you specify either in the PERFORMANCE statement or in the grid environment variables. For more information about distributed processing modes, see the section "Processing Modes" on page 6.

In the client-data model, PROC HPSEVERITY distributes the input data across the number of nodes that you specify by sending the first observation to the first node, the second observation to the second node, and so on.

In the alongside-the-database model, PROC HPSEVERITY uses the existing distributed organization of the data. You do not need to specify the NODES= option.

The number of nodes that are used for distributed computing is displayed in the "Performance Information" table, which is part of the default output.

Multithreading

Threading refers to the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading refers to the concurrent execution of threads. When multithreading is possible, you can achieve more substantial performance gains than you can with sequential (single-threaded) execution.

The number of threads the HPSEVERITY procedure spawns is determined by the number of CPUs on a machine. You can control the number of CPUs in the following ways:

- You can use the CPUCOUNT= SAS system option to specify the CPU count. For example, if you specify the following statement, then PROC HPSEVERITY schedules threads as if it were executing on a system that had four CPUs, regardless of the actual CPU count:

  ```
  options cpucount=4;
  ```

  You can use this specification only in single-machine mode, and it does not take effect if the THREADS system option is turned off.

  The default value of the CPUCOUNT= system option might not equal the number of all the logical CPU cores available on your machine, such as those available because of hyperthreading. To allow PROC HPSEVERITY to use all the logical cores in single-machine mode, specify the following OPTIONS statement:

  ```
  options cpucount=actual;
  ```

- You can specify the NTHREADS= option in the PERFORMANCE statement. This specification overrides the THREADS and CPUCOUNT= system options. Specify NTHREADS=1 to force single-threaded execution.

If you do not specify the NTHREADS= option and the THREADS system option is turned on, then the number of threads that are used in distributed mode is equal to the total number of logical CPU cores available on each node of the grid, and the number of threads used in single-machine mode is determined by the CPUCOUNT= system option.

If you do not specify the NTHREADS= option and the THREADS system option is turned off, then only one thread of execution is used in both single-machine and distributed modes.

The number of threads per machine is displayed in the "Performance Information" table, which is part of the default output.

Performance improvement is not always guaranteed when you use more threads, for several reasons: the increased cost of communication and synchronization among threads might offset the reduced cost of computation, the hyperthreading feature of the processor might not be very efficient for floating-point computations, and other applications might be running on the machine.

Combining the Power of Distributed and Multithreading Computing

The HPSEVERITY procedure combines the powers of distributed and multithreading paradigms by using a data-parallel model. In particular, the distributed tasks are defined by dividing the data among multiple nodes, and within one node, the multithreading tasks are defined by further dividing the local data among the threads. For example, if the input data set has 10,000 observations and you are running on a grid that has five nodes, then each node processes 2,000 observations (this assumes that if you have specified an alongside-the-database model, then you have equally and randomly divided the input data among the nodes). Further, if each node has eight CPUs, then 250 observations are associated with each thread within the node. All computations that require access to the data are then distributed and multithreaded.

Note that in single-machine mode (see the section "Processing Modes" on page 6), only multithreading is available.

When you specify more than one candidate distribution model, for some tasks PROC HPSEVERITY exploits the independence among models by processing multiple models in parallel on a single node such that each model is assigned to one of the threads executing in parallel. When a thread finishes processing the assigned model, it starts processing the next unprocessed model, if one exists.

The computations that take advantage of the distributed and multithreaded model include the following:

- Validation and preparation of data: In this stage, the observations in the input data set are validated and transformed, if necessary. The summary statistics of the data are prepared. Because each observation is independent, the computations can be distributed among nodes and among threads within nodes without significant communication overhead.

- Initialization of distribution parameters: In this stage, the parallelism is achieved by initializing multiple models in parallel. The only computational step that is not fully parallelized in this release is the step of computing empirical distribution function (EDF) estimates, which are required when PROC HPSEVERITY needs to invoke a distribution's PARMINIT subroutine to initialize distribution parameters. The EDF estimation step is not amenable to full-fledged parallelism because it requires sequential access to sorted data, especially when the loss variable is modified by truncation effects. When the data are distributed across nodes, the EDF computations take place on local data and the PARMINIT function is invoked on the local data by using the local EDF estimates. The initial values that are supplied to the nonlinear optimizer are computed by averaging the local estimates of the distribution parameters that are returned by the PARMINIT functions on each node.

- Initialization of regression parameters (if SCALEMODEL statement is specified): In this stage, if you have not specified initial values for the regression parameters by using the INEST= data set, then PROC HPSEVERITY initializes those parameters by solving a linear regression problem $\log(y) = \beta_0 + \sum_{i=1}^{k} \beta_j x_j$. For more information, see the section "Parameter Initialization for Regression Models" on page 176. The most computationally intensive step is the formation of the crossproducts matrix. PROC HPSEVERITY exploits the parallelism by observing the fact that the contribution to the crossproducts matrix due to one observation is independent from the contribution due to another observation. Each node computes the contribution of its local data to each entry of the crossproducts matrix. Within each node, each thread computes the contribution of its chunk of data to each entry of the crossproducts matrix. On each node, the contributions from all the threads are added up to form the contribution due to all of the local data. The partial crossproducts matrices are then gathered from all nodes on a central node, which sums them up to form the final crossproducts matrix.

- Optimization: In this stage, the nonlinear optimizer iterates over the parameter space in search of the optimal set of parameters. In each iteration, it evaluates the objective function along with the gradient and Hessian of the objective function, if needed by the optimization method. Within one iteration, for the current estimates of the parameters, each observation's contribution to the objective function, gradient, and Hessian is independent of another observation. This enables PROC HPSEVERITY to fully exploit the distributed and multithreaded paradigms to efficiently parallelize each iteration of the algorithm.

Defining a Severity Distribution Model with the FCMP Procedure

A *severity distribution model* consists of a set of functions and subroutines that are defined using the FCMP procedure. The FCMP procedure is part of Base SAS software. Each function or subroutine must be named as <*distribution-name*>_<*keyword*>, where *distribution-name* is the identifying short name of the distribution and *keyword* identifies one of the functions or subroutines. The total length of the name should not exceed 32. Each function or subroutine must have a specific signature, which consists of the number of arguments, sequence and types of arguments, and return value type. The summary of all the recognized function and subroutine names and their expected behavior is given in Table 5.6.

Consider the following points when you define a distribution model:

- When you define a function or subroutine requiring parameter arguments, the names and order of those arguments must be the same. Arguments other than the parameter arguments can have any name, but they must satisfy the requirements on their type and order.

- When the HPSEVERITY procedure invokes any function or subroutine, it provides the necessary input values according to the specified signature, and expects the function or subroutine to prepare the output and return it according to the specification of the return values in the signature.

- You can use most of the SAS programming statements and SAS functions that you can use in a DATA step for defining the FCMP functions and subroutines. However, there are a few differences in the capabilities of the DATA step and the FCMP procedure. To learn more, see the documentation of the FCMP procedure in the *Base SAS Procedures Guide*.

- You must specify either the PDF or the LOGPDF function. Similarly, you must specify either the CDF or the LOGCDF function. All other functions are optional, except when necessary for correct definition of the distribution. It is strongly recommended that you define the PARMINIT subroutine to provide a good set of initial values for the parameters. The information provided by PROC HPSEVERITY to the PARMINIT subroutine enables you to use popular initialization approaches based on the method of moments and the method of percentile matching, but you can implement any algorithm to initialize the parameters by using the values of the response variable and the estimate of its empirical distribution function.

- The LOWERBOUNDS subroutines should be defined if the lower bound on at least one distribution parameter is different from the default lower bound of 0. If you define a LOWERBOUNDS subroutine but do not set a lower bound for some parameter inside the subroutine, then that parameter is assumed to have no lower bound (or a lower bound of $-\infty$). Hence, it is recommended that you explicitly return the lower bound for each parameter when you define the LOWERBOUNDS subroutine.

- The UPPERBOUNDS subroutines should be defined if the upper bound on at least one distribution parameter is different from the default upper bound of ∞. If you define an UPPERBOUNDS subroutine but do not set an upper bound for some parameter inside the subroutine, then that parameter is assumed to have no upper bound (or a upper bound of ∞). Hence, it is recommended that you explicitly return the upper bound for each parameter when you define the UPPERBOUNDS subroutine.

- If you want to use the distribution in a model with regression effects, then make sure that the first parameter of the distribution is the scale parameter itself or a log-transformed scale parameter. If the first parameter is a log-transformed scale parameter, then you must define the SCALETRANSFORM function.

- In general, it is not necessary to define the gradient and Hessian functions, because the HPSEVERITY procedure uses an internal system to evaluate the required derivatives. The internal system typically computes the derivatives analytically. But it might not be able to do so if your function definitions use other functions that it cannot differentiate analytically. In such cases, derivatives are approximated using a finite difference method and a note is written to the SAS log to indicate the components that are differentiated using such approximations. PROC HPSEVERITY does reasonably well with these finite difference approximations. But, if you know of a way to compute the derivatives of such components analytically, then you should define the gradient and Hessian functions.

In order to use your distribution with PROC HPSEVERITY, you need to record the FCMP library that contains the functions and subroutines for your distribution and other FCMP libraries that contain FCMP functions or subroutines used within your distribution's functions and subroutines. Specify all those libraries in the CMPLIB= system option by using the OPTIONS global statement. For more information about the OPTIONS statement, see *SAS Statements: Reference*. For more information about the CMPLIB= system option, see *SAS System Options: Reference*.

Each predefined distribution mentioned in the section "Predefined Distributions" on page 159 has a distribution model associated with it. The functions and subroutines of all those models are available in the Sashelp.Svrtdist library. The order of the parameters in the signatures of the functions and subroutines is the same as listed in Table 5.2. You do not need to use the CMPLIB= option in order to use the predefined distributions with PROC HPSEVERITY. However, if you need to use the functions or subroutines of the predefined distributions in SAS statements other than the PROC HPSEVERITY step (such as in a DATA step), then specify the Sashelp.Svrtdist library in the CMPLIB= system option by using the OPTIONS global statement prior to using them.

Table 5.6 shows functions and subroutines that define a distribution model, and subsections after the table provide more detail. The functions are listed in alphabetical order of the keyword suffix.

Table 5.6 List of Functions and Subroutines That Define a Distribution Model

Name	**Type**	**Required**	**Expected to Return**
*dist*_CDF	Function	YES[1]	Cumulative distribution function value
*dist*_CDFGRADIENT	Subroutine	NO	Gradient of the CDF
*dist*_CDFHESSIAN	Subroutine	NO	Hessian of the CDF
*dist*_CONSTANTPARM	Subroutine	NO	Constant parameters
*dist*_DESCRIPTION	Function	NO	Description of the distribution
*dist*_LOGCDF	Function	YES[1]	Log of cumulative distribution function value
*dist*_LOGCDFGRADIENT	Subroutine	NO	Gradient of the LOGCDF
*dist*_LOGCDFHESSIAN	Subroutine	NO	Hessian of the LOGCDF
*dist*_LOGPDF	Function	YES[2]	Log of probability density function value
*dist*_LOGPDFGRADIENT	Subroutine	NO	Gradient of the LOGPDF
*dist*_LOGPDFHESSIAN	Subroutine	NO	Hessian of the LOGPDF
*dist*_LOGSDF	Function	NO	Log of survival function value
*dist*_LOGSDFGRADIENT	Subroutine	NO	Gradient of the LOGSDF
*dist*_LOGSDFHESSIAN	Subroutine	NO	Hessian of the LOGSDF
*dist*_LOWERBOUNDS	Subroutine	NO	Lower bounds on parameters
*dist*_PARMINIT	Subroutine	NO	Initial values for parameters
*dist*_PDF	Function	YES[2]	Probability density function value
*dist*_PDFGRADIENT	Subroutine	NO	Gradient of the PDF
*dist*_PDFHESSIAN	Subroutine	NO	Hessian of the PDF
*dist*_SCALETRANSFORM	Function	NO	Type of relationship between the first distribution parameter and the scale parameter
*dist*_SDF	Function	NO	Survival function value
*dist*_SDFGRADIENT	Subroutine	NO	Gradient of the SDF
*dist*_SDFHESSIAN	Subroutine	NO	Hessian of the SDF
*dist*_UPPERBOUNDS	Subroutine	NO	Upper bounds on parameters

Notes:
1. Either the *dist*_CDF or the *dist*_LOGCDF function must be defined.
2. Either the *dist*_PDF or the *dist*_LOGPDF function must be defined.

The signature syntax and semantics of each function or subroutine are as follows:

*dist*_**CDF**

defines a function that returns the value of the cumulative distribution function (CDF) of the distribution at the specified values of the random variable and distribution parameters.

- *Type*: Function
- *Required*: YES
- *Number of arguments*: $m + 1$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

x	Numeric value of the random variable at which the CDF value should be evaluated
p1	Numeric value of the first parameter
p2	Numeric value of the second parameter
	. . . ..
pm	Numeric value of the mth parameter

- *Return value*: Numeric value that contains the CDF value $F(x; p_1, p_2, \ldots, p_m)$

If you want to consider this distribution as a candidate distribution when you estimate a response variable model with regression effects, then the first parameter of this distribution must be a scale parameter or log-transformed scale parameter. In other words, if the distribution has a scale parameter, then the following equation must be satisfied:

$$F(x; p_1, p_2, \ldots, p_m) = F(\frac{x}{p_1}; 1, p_2, \ldots, p_m)$$

If the distribution has a log-transformed scale parameter, then the following equation must be satisfied:

$$F(x; p_1, p_2, \ldots, p_m) = F(\frac{x}{\exp(p_1)}; 0, p_2, \ldots, p_m)$$

Here is a sample structure of the function for a distribution named 'FOO':

```
function FOO_CDF(x, P1, P2);
   /* Code to compute CDF by using x, P1, and P2 */

   F = <computed CDF>;
   return (F);
endsub;
```

*dist*_**CONSTANTPARM**

defines a subroutine that specifies constant parameters. A parameter is *constant* if it is required for defining a distribution but is not subject to optimization in PROC HPSEVERITY. Constant parameters are required to be part of the model in order to compute the PDF or the CDF of the distribution. Typically, values of these parameters are known a priori or estimated using some means other than the maximum likelihood method used by PROC HPSEVERITY. You can estimate them inside the *dist*_PARMINIT subroutine. Once initialized, the parameters remain constant in the context of PROC HPSEVERITY; that is, they retain their initial value. PROC HPSEVERITY estimates only the nonconstant parameters.

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: k, where k is the number of constant parameters

- *Sequence and type of arguments*:

constant parameter 1	Name of the first constant parameter
.....	
constant parameter k	Name of the kth constant parameter

- *Return value*: None

Here is a sample structure of the subroutine for a distribution named ‘FOO’ that has P3 and P5 as its constant parameters, assuming that distribution has at least three parameters:

```
subroutine FOO_CONSTANTPARM(p5, p3);
endsub;
```

Note the following points when you specify the constant parameters:

- At least one distribution parameter must be free to be optimized; that is, if a distribution has total m parameters, then k must be strictly less than m.
- If you want to use this distribution for modeling regression effects, then the first parameter must not be a constant parameter.
- The order of arguments in the signature of this subroutine does not matter as long as each argument’s name matches the name of one of the parameters that are defined in the signature of the *dist*_PDF function.
- The constant parameters must be specified in signatures of all the functions and subroutines that accept distribution parameters as their arguments.
- You must provide a nonmissing initial value for each constant parameter by using one of the supported parameter initialization methods.

***dist*_DESCRIPTION**

defines a function that returns a description of the distribution.

- *Type*: Function
- *Required*: NO
- *Number of arguments*: None
- *Sequence and type of arguments*: Not applicable
- *Return value*: Character value containing a description of the distribution

Here is a sample structure of the function for a distribution named ‘FOO’:

```
function FOO_DESCRIPTION() $48;
    length desc $48;
    desc = "A model for a continuous distribution named foo";
    return (desc);
endsub;
```

There is no restriction on the length of the description (the length of 48 used in the previous example is for illustration purposes only). However, if the length is greater than 256, then only the first 256 characters appear in the displayed output and in the _DESCRIPTION_ variable of the OUTMODELINFO= data set. Hence, the recommended length of the description is less than or equal to 256.

*dist*_**LOG***core*

defines a function that returns the natural logarithm of the specified *core* function of the distribution at the specified values of the random variable and distribution parameters. The *core* keyword can be PDF, CDF, or SDF.

- *Type*: Function
- *Required*: YES only if *core* is PDF or CDF and you have not defined that *core* function; otherwise, NO
- *Number of arguments*: $m + 1$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

x	Numeric value of the random variable at which the natural logarithm of the *core* function should be evaluated
p1	Numeric value of the first parameter
p2	Numeric value of the second parameter
	. . . ..
pm	Numeric value of the mth parameter

- *Return value*: Numeric value that contains the natural logarithm of the *core* function

Here is a sample structure of the function for the core function PDF of a distribution named 'FOO':

```
function FOO_LOGPDF(x, P1, P2);
   /* Code to compute LOGPDF by using x, P1, and P2 */

   l = <computed LOGPDF>;
   return (l);
endsub;
```

*dist*_**LOWERBOUNDS**

defines a subroutine that returns lower bounds for the parameters of the distribution. If this subroutine is not defined for a given distribution, then the HPSEVERITY procedure assumes a lower bound of 0 for each parameter. If a lower bound of l_i is returned for a parameter p_i, then the HPSEVERITY procedure assumes that $l_i < p_i$ (strict inequality). If a missing value is returned for some parameter, then the HPSEVERITY procedure assumes that there is no lower bound for that parameter (equivalent to a lower bound of $-\infty$).

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: m, where m is the number of distribution parameters
- *Sequence and type of arguments*:

p1	Output argument that returns the lower bound on the first parameter. This must be specified in the OUTARGS statement inside the subroutine's definition.
p2	Output argument that returns the lower bound on the second parameter. This must be specified in the OUTARGS statement inside the subroutine's definition.
	. . . ..

p*m*	Output argument that returns the lower bound on the *m*th parameter. This must be specified in the OUTARGS statement inside the subroutine's definition.

- *Return value*: The results, lower bounds on parameter values, should be returned in the parameter arguments of the subroutine.

Here is a sample structure of the subroutine for a distribution named 'FOO':

```
subroutine FOO_LOWERBOUNDS(p1, p2);
    outargs p1, p2;

    p1 = <lower bound for P1>;
    p2 = <lower bound for P2>;
endsub;
```

*dist*_**PARMINIT**

defines a subroutine that returns the initial values for the distribution's parameters given an empirical distribution function (EDF) estimate.

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: $m + 4$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

dim	Input numeric value that contains the dimension of the x, nx, and F array arguments.
x{*}	Input numeric array of dimension *dim* that contains values of the random variables at which the EDF estimate is available. It can be assumed that x contains values in an increasing order. In other words, if $i < j$, then x[i] < x[j].
nx{*}	Input numeric array of dimension *dim*. Each nx[i] contains the number of observations in the original data that have the value x[i].
F{*}	Input numeric array of dimension *dim*. Each F[i] contains the EDF estimate for x[i]. This estimate is computed by the HPSEVERITY procedure based on the options specified in the LOSS statement.
Ftype	Input numeric value that contains the type of the EDF estimate that is stored in x and F. See the section "Supplying EDF Estimates to Functions and Subroutines" on page 184 for definition of types.
p1	Output argument that returns the initial value of the first parameter. This must be specified in the OUTARGS statement inside the subroutine's definition.
p2	Output argument that returns the initial value of the second parameter. This must be specified in the OUTARGS statement inside the subroutine's definition.
	. . . ..
p*m*	Output argument that returns the initial value of the *m*th parameter. This must be specified in the OUTARGS statement inside the subroutine's definition.

- *Return value*: The results, initial values of the parameters, should be returned in the parameter arguments of the subroutine.

Here is a sample structure of the subroutine for a distribution named ‘FOO’:

```
subroutine FOO_PARMINIT(dim, x{*}, nx{*}, F{*}, Ftype, p1, p2);
    outargs p1, p2;

    /* Code to initialize values of P1 and P2 by using
       dim, x, nx, and F */

    p1 = <initial value for p1>;
    p2 = <initial value for p2>;
endsub;
```

*dist*_**PDF**

defines a function that returns the value of the probability density function (PDF) of the distribution at the specified values of the random variable and distribution parameters.

- *Type*: Function
- *Required*: YES
- *Number of arguments*: $m + 1$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

x	Numeric value of the random variable at which the PDF value should be evaluated
p1	Numeric value of the first parameter
p2	Numeric value of the second parameter
	.....
pm	Numeric value of the mth parameter

- *Return value*: Numeric value that contains the PDF value $f(x; p_1, p_2, \dots, p_m)$

If you want to consider this distribution as a candidate distribution when you estimate a response variable model with regression effects, then the first parameter of this distribution must be a scale parameter or log-transformed scale parameter. In other words, if the distribution has a scale parameter, then the following equation must be satisfied:

$$f(x; p_1, p_2, \dots, p_m) = \frac{1}{p_1} f(\frac{x}{p_1}; 1, p_2, \dots, p_m)$$

If the distribution has a log-transformed scale parameter, then the following equation must be satisfied:

$$f(x; p_1, p_2, \dots, p_m) = \frac{1}{\exp(p_1)} f(\frac{x}{\exp(p_1)}; 0, p_2, \dots, p_m)$$

Here is a sample structure of the function for a distribution named ‘FOO’:

```
function FOO_PDF(x, P1, P2);
    /* Code to compute PDF by using x, P1, and P2 */

    f = <computed PDF>;
    return (f);
endsub;
```

***dist*_SCALETRANSFORM**

defines a function that returns a keyword to identify the transform that needs to be applied to the scale parameter to convert it to the first parameter of the distribution.

If you want to use this distribution for modeling regression effects, then the first parameter of this distribution must be a scale parameter. However, for some distributions, a typical or convenient parameterization might not have a scale parameter, but one of the parameters can be a simple transform of the scale parameter. As an example, consider a typical parameterization of the lognormal distribution with two parameters, location μ and shape σ, for which the PDF is defined as follows:

$$f(x; \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\log(x)-\mu}{\sigma}\right)^2}$$

You can reparameterize this distribution to contain a parameter θ instead of the parameter μ such that $\mu = \log(\theta)$. The parameter θ would then be a scale parameter. However, if you want to specify the distribution in terms of μ and σ (which is a more recognized form of the lognormal distribution) and still allow it as a candidate distribution for estimating regression effects, then instead of writing another distribution with parameters θ and σ, you can simply define the distribution with μ as the first parameter and specify that it is the logarithm of the scale parameter.

- *Type*: Function
- *Required*: NO
- *Number of arguments*: None
- *Sequence and type of arguments*: Not applicable
- *Return value*: Character value that contains one of the following keywords:

LOG	specifies that the first parameter is the logarithm of the scale parameter.
IDENTITY	specifies that the first parameter is a scale parameter without any transformation.

If this function is not specified, then the IDENTITY transform is assumed.

Here is a sample structure of the function for a distribution named ‘FOO’:

```
function FOO_SCALETRANSFORM() $8;
    length xform $8;
    xform = "IDENTITY";
    return (xform);
endsub;
```

***dist*_SDF**

defines a function that returns the value of the survival distribution function (SDF) of the distribution at the specified values of the random variable and distribution parameters.

- *Type*: Function
- *Required*: NO
- *Number of arguments*: $m + 1$, where m is the number of distribution parameters

- *Sequence and type of arguments*:

  x Numeric value of the random variable at which the SDF value should be evaluated

  p1 Numeric value of the first parameter

  p2 Numeric value of the second parameter

  .....

  pm Numeric value of the mth parameter

- *Return value*: Numeric value that contains the SDF value $S(x; p_1, p_2, \ldots, p_m)$

If you want to consider this distribution as a candidate distribution when estimating a response variable model with regression effects, then the first parameter of this distribution must be a scale parameter or log-transformed scale parameter. In other words, if the distribution has a scale parameter, then the following equation must be satisfied:

$$S(x; p_1, p_2, \ldots, p_m) = S(\frac{x}{p_1}; 1, p_2, \ldots, p_m)$$

If the distribution has a log-transformed scale parameter, then the following equation must be satisfied:

$$S(x; p_1, p_2, \ldots, p_m) = S(\frac{x}{\exp(p_1)}; 0, p_2, \ldots, p_m)$$

Here is a sample structure of the function for a distribution named 'FOO':

```
function FOO_SDF(x, P1, P2);
   /* Code to compute SDF by using x, P1, and P2 */

   S = <computed SDF>;
   return (S);
endsub;
```

*dist*_**UPPERBOUNDS**

defines a subroutine that returns upper bounds for the parameters of the distribution. If this subroutine is not defined for a given distribution, then the HPSEVERITY procedure assumes that there is no upper bound for any of the parameters. If an upper bound of u_i is returned for a parameter p_i, then the HPSEVERITY procedure assumes that $p_i < u_i$ (strict inequality). If a missing value is returned for some parameter, then the HPSEVERITY procedure assumes that there is no upper bound for that parameter (equivalent to an upper bound of ∞).

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: m, where m is the number of distribution parameters
- *Sequence and type of arguments*:

  p1 Output argument that returns the upper bound on the first parameter. This must be specified in the OUTARGS statement inside the subroutine's definition.

  p2 Output argument that returns the upper bound on the second parameter. This must be specified in the OUTARGS statement inside the subroutine's definition.

  .....

p*m*	Output argument that returns the upper bound on the *m*th parameter. This must be specified in the OUTARGS statement inside the subroutine's definition.

- *Return value*: The results, upper bounds on parameter values, should be returned in the parameter arguments of the subroutine.

Here is a sample structure of the subroutine for a distribution named 'FOO':

```
subroutine FOO_UPPERBOUNDS(p1, p2);
    outargs p1, p2;

    p1 = <upper bound for P1>;
    p2 = <upper bound for P2>;
endsub;
```

*dist_core*GRADIENT

defines a subroutine that returns the gradient vector of the specified *core* function of the distribution at the specified values of the random variable and distribution parameters. The *core* keyword can be PDF, CDF, SDF, LOGPDF, LOGCDF, or LOGSDF.

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: $m + 2$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

x	Numeric value of the random variable at which the gradient should be evaluated
p1	Numeric value of the first parameter
p2	Numeric value of the second parameter
	. . . ..
p*m*	Numeric value of the *m*th parameter
grad{*}	Output numeric array of size m that contains the gradient vector evaluated at the specified values. If h denotes the value of the *core* function, then the expected order of the values in the array is as follows: $\frac{\partial h}{\partial p_1}\ \frac{\partial h}{\partial p_2} \cdots \frac{\partial h}{\partial p_m}$

- *Return value*: Numeric array that contains the gradient evaluated at x for the parameter values $(p_1, p_2, \ldots, p_m)$

Here is a sample structure of the function for the core function CDF of a distribution named 'FOO':

```
subroutine FOO_CDFGRADIENT(x, P1, P2, grad{*});
    outargs grad;

    /* Code to compute gradient by using x, P1, and P2 */
    grad[1] = <partial derivative of CDF w.r.t. P1
                  evaluated at x, P1, P2>;
    grad[2] = <partial derivative of CDF w.r.t. P2
                  evaluated at x, P1, P2>;
endsub;
```

*dist_core***HESSIAN**

defines a subroutine that returns the Hessian matrix of the specified *core* function of the distribution at the specified values of the random variable and distribution parameters. The *core* keyword can be PDF, CDF, SDF, LOGPDF, LOGCDF, or LOGSDF.

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: $m + 2$, where m is the number of distribution parameters
- *Sequence and type of arguments*:

x	Numeric value of the random variable at which the Hessian matrix should be evaluated
p1	Numeric value of the first parameter
p2	Numeric value of the second parameter
	…..
pm	Numeric value of the mth parameter
hess{*}	Output numeric array of size $m(m+1)/2$ that contains the lower triangular portion of the Hessian matrix in a packed vector form, evaluated at the specified values. If h denotes the value of the *core* function, then the expected order of the values in the array is as follows: $\frac{\partial^2 h}{\partial p_1^2} \mid \frac{\partial^2 h}{\partial p_1 \partial p_2} \; \frac{\partial^2 h}{\partial p_2^2} \mid \cdots \mid \frac{\partial^2 h}{\partial p_1 \partial p_m} \; \frac{\partial^2 h}{\partial p_2 \partial p_m} \cdots \frac{\partial^2 h}{\partial p_m^2}$

- *Return value*: Numeric array that contains the lower triangular portion of the Hessian matrix evaluated at x for the parameter values $(p_1, p_2, \ldots, p_m)$

Here is a sample structure of the subroutine for the core function LOGSDF of a distribution named 'FOO':

```
subroutine FOO_LOGSDFHESSIAN(x, P1, P2, hess{*});
    outargs hess;

    /* Code to compute Hessian by using x, P1, and P2 */
    hess[1] = <second order partial derivative of LOGSDF
                 w.r.t. P1 evaluated at x, P1, P2>;
    hess[2] = <second order partial derivative of LOGSDF
                 w.r.t. P1 and P2 evaluated at x, P1, P2>;
    hess[3] = <second order partial derivative of LOGSDF
                 w.r.t. P2 evaluated at x, P1, P2>;
endsub;
```

Predefined Utility Functions

The following predefined utility functions are provided with the HPSEVERITY procedure and are available in the Sashelp.Svrtdist library:

SVRTUTIL_EDF:

This function computes the empirical distribution function (EDF) estimate at the specified value of the random variable given the EDF estimate for a sample.

- *Type*: Function
- *Signature*: SVRTUTIL_EDF(y, n, x{*}, F{*}, Ftype)
- *Argument Description*:

y	Value of the random variable at which the EDF estimate is desired.
n	Dimension of the *x* and *F* input arrays.
x{*}	Input numeric array of dimension *n* that contains values of the random variable observed in the sample. These values are sorted in nondecreasing order.
F{*}	Input numeric array of dimension *n* in which each F[*i*] contains the EDF estimate for x[*i*]. These values must be sorted in nondecreasing order.
Ftype	Type of the empirical estimate that is stored in the *x* and *F* arrays. See the section "Supplying EDF Estimates to Functions and Subroutines" on page 184 for definition of types.

- *Return value*: The EDF estimate at y.

The type of the sample EDF estimate determines how the EDF estimate at y is computed. For more information, see the section "Supplying EDF Estimates to Functions and Subroutines" on page 184.

SVRTUTIL_EMPLIMMOMENT:

This function computes the empirical estimate of the limited moment of specified order for the specified upper limit, given the EDF estimate for a sample.

- *Type*: Function
- *Signature*: SVRTUTIL_EMPLIMMOMENT(k, u, n, x{*}, F{*}, Ftype)
- *Argument Description*:

k	Order of the desired empirical limited moment.
u	Upper limit on the value of the random variable to be used in the computation of the desired empirical limited moment.
n	Dimension of the *x* and *F* input arrays.
x{*}	Input numeric array of dimension *n* that contains values of the random variable observed in the sample. These values are sorted in nondecreasing order.
F{*}	Input numeric array of dimension *n* in which each F[*i*] contains the EDF estimate for x[*i*]. These values must be sorted in nondecreasing order.

Ftype	Type of the empirical estimate that is stored in the x and F arrays. See the section "Supplying EDF Estimates to Functions and Subroutines" on page 184 for definition of types.

- *Return value*: The desired empirical limited moment.

The empirical limited moment is computed by using the empirical estimate of the CDF. If $F_n(x)$ denotes the EDF at x, then the empirical limited moment of order k with upper limit u is defined as

$$E_n[(X \wedge u)^k] = k \int_0^u (1 - F_n(x))x^{k-1} dx$$

The SVRTUTIL_EMPLIMMOMENT function uses the piecewise linear nature of $F_n(x)$ as described in the section "Supplying EDF Estimates to Functions and Subroutines" on page 184 to compute the integration.

SVRTUTIL_HILLCUTOFF:

This function computes an estimate of the value where the right tail of a distribution is expected to begin. The function implements the algorithm described in Danielsson et al. 2001. The description of the algorithm uses the following notation:

n	Number of observations in the original sample.
B	Number of bootstrap samples to draw.
m_1	Size of the bootstrap sample in the first step of the algorithm ($m_1 < n$).
$x_{(i)}^{j,m}$	ith order statistic of jth bootstrap sample of size m ($1 \leq i \leq m, 1 \leq j \leq B$).
$x_{(i)}$	ith order statistic of the original sample ($1 \leq i \leq n$).

Given the input sample x and values of B and m_1, the steps of the algorithm are as follows:

1. Take B bootstrap samples of size m_1 from the original sample.
2. Find the integer k_1 that minimizes the bootstrap estimate of the mean squared error:

$$k_1 = \arg \min_{1 \leq k < m_1} Q(m_1, k)$$

3. Take B bootstrap samples of size $m_2 = m_1^2/n$ from the original sample.
4. Find the integer k_2 that minimizes the bootstrap estimate of the mean squared error:

$$k_2 = \arg \min_{1 \leq k < m_2} Q(m_2, k)$$

5. Compute the integer k_{opt}, which is used for computing the cutoff point:

$$k_{\text{opt}} = \frac{k_1^2}{k_2} \left(\frac{\log(k_1)}{2\log(m_1) - \log(k_1)} \right)^{2-2\log(k_1)/\log(m_1)}$$

6. Set the cutoff point equal to $x_{(k_{\text{opt}}+1)}$.

The bootstrap estimate of the mean squared error is computed as

$$Q(m,k) = \frac{1}{B}\sum_{j=1}^{B} \mathrm{MSE}_j(m,k)$$

The mean squared error of jth bootstrap sample is computed as

$$\mathrm{MSE}_j(m,k) = (M_j(m,k) - 2(\gamma_j(m,k))^2)^2$$

where $M_j(m,k)$ is a control variate proposed by Danielsson et al. 2001,

$$M_j(m,k) = \frac{1}{k}\sum_{i=1}^{k} \left(\log(x^{j,m}_{(m-i+1)}) - \log(x^{j,m}_{(m-k)})\right)^2$$

and $\gamma_j(m,k)$ is the Hill's estimator of the tail index (Hill 1975),

$$\gamma_j(m,k) = \frac{1}{k}\sum_{i=1}^{k} \log(x^{j,m}_{(m-i+1)}) - \log(x^{j,m}_{(m-k)})$$

This algorithm has two tuning parameters, B and m_1. The number of bootstrap samples B is chosen based on the availability of computational resources. The optimal value of m_1 is chosen such that the following ratio, $R(m_1)$, is minimized:

$$R(m_1) = \frac{(Q(m_1,k_1))^2}{Q(m_2,k_2)}$$

The SVRTUTIL_HILLCUTOFF utility function implements the preceding algorithm. It uses the grid search method to compute the optimal value of m_1.

- *Type*: Function
- *Signature*: SVRTUTIL_HILLCUTOFF(n, x{*}, b, s, status)
- *Argument Description*:

n	Dimension of the array *x*.
x{*}	Input numeric array of dimension n that contains the sample.
b	Number of bootstrap samples used to estimate the mean squared error. If *b* is less than 10, then a default value of 50 is used.
s	Approximate number of steps used to search the optimal value of m_1 in the range $[n^{0.75}, n-1]$. If *s* is less than or equal to 1, then a default value of 10 is used.
status	Output argument that contains the status of the algorithm. If the algorithm succeeds in computing a valid cutoff point, then *status* is set to 0. If the algorithm fails, then *status* is set to 1.

- *Return value*: The cutoff value where the right tail is estimated to start. If the size of the input sample is inadequate ($n \leq 5$), then a missing value is returned and *status* is set to a missing value. If the algorithm fails to estimate a valid cutoff value (*status* = 1), then the fifth largest value in the input sample is returned.

SVRTUTIL_PERCENTILE:

This function computes the specified empirical percentile given the EDF estimates.

- *Type*: Function
- *Signature*: SVRTUTIL_PERCENTILE(p, n, x{*}, F{*}, Ftype)
- *Argument Description*:

p	Desired percentile. The value must be in the interval (0,1). The function returns the $100p$th percentile.
n	Dimension of the x and F input arrays.
x{*}	Input numeric array of dimension n that contains values of the random variable observed in the sample. These values are sorted in nondecreasing order.
F{*}	Input numeric array of dimension n in which each F[i] contains the EDF estimate for x[i]. These values must be sorted in nondecreasing order.
Ftype	Type of the empirical estimate that is stored in the x and F arrays. See the section "Supplying EDF Estimates to Functions and Subroutines" on page 184 for definition of types.

- *Return value*: The $100p$th percentile of the input sample.

The method used to compute the percentile depends on the type of the EDF estimate (Ftype argument).

Ftype = 1

Smoothed empirical estimates are computed using the method described in Klugman, Panjer, and Willmot (1998). Let $\lfloor x \rfloor$ denote the greatest integer less than or equal to x. Define $g = \lfloor p(n+1) \rfloor$ and $h = p(n+1) - g$. Then the empirical percentile $\hat{\pi}_p$ is defined as

$$\hat{\pi}_p = (1-h)x[g] + hx[g+1]$$

This method does not work if $p < 1/(n+1)$ or $p > n/(n+1)$. If $p < 1/(n+1)$, then the function returns $\hat{\pi}_p = x[1]/2$, which assumes that the EDF is 0 in the interval $[0, x[1])$. If $p > n/(n+1)$, then $\hat{\pi}_p = x[n]$.

Ftype = 2

If $p < F[1]$, then $\hat{\pi}_p = x[1]/2$, which assumes that the EDF is 0 in the interval $[0, x[1])$. If $|p - F[i]| < \epsilon$ for some value of i and $i < n$, then $\hat{\pi}_p$ is computed as

$$\hat{\pi}_p = \frac{x[i] + x[i+1]}{2}$$

where ϵ is a machine-precision constant as returned by the SAS function CONSTANT('MACEPS'). If $F[i-1] < p < F[i]$, then $\hat{\pi}_p$ is computed as

$$\hat{\pi}_p = x[i]$$

If $p \geq F[n]$, then $\hat{\pi}_p = x[n]$.

SVRTUTIL_RAWMOMENTS:

This subroutine computes the raw moments of a sample.

- *Type*: Subroutine
- *Signature*: SVRTUTIL_RAWMOMENTS(n, x{*}, nx{*}, nRaw, raw{*})

- *Argument Description*:

n	Dimension of the *x* and *nx* input arrays.
x{*}	Input numeric array of dimension *n* that contains distinct values of the random variable that are observed in the sample.
nx{*}	Input numeric array of dimension *n* in which each *nx*[i] contains the number of observations in the sample that have the value x[i].
nRaw	Desired number of raw moments. The output array *raw* contains the first *nRaw* raw moments.
raw{*}	Output array of raw moments. The kth element in the array (raw{k}) contains the kth raw moment, where $1 \leq k \leq$ nRaw.

- *Return value*: Numeric array *raw* that contains the first *nRaw* raw moments. The array contains missing values if the sample has no observations (that is, if all the values in the *nx* array add up to zero).

SVRTUTIL_SORT:
This function sorts the given array of numeric values in an ascending or descending order.

- *Type*: Subroutine
- *Signature*: SVRTUTIL_SORT(n, x{*}, flag)
- *Argument Description*:

n	Dimension of the input array *x*.
x{*}	Numeric array that contains the values to be sorted at input. The subroutine uses the same array to return the sorted values.
flag	A numeric value that controls the sort order. If *flag* is 0, then the values are sorted in an ascending order. If *flag* has any value other than 0, then the values are sorted in descending order.

- *Return value*: Numeric array *x*, which is sorted in place (that is, the sorted array is stored in the same storage area occupied by the input array *x*).

You can use the following predefined functions when you use the FCMP procedure to define functions and subroutines. They are summarized here for your information. For more information, see the FCMP procedure documentation in *Base SAS Procedures Guide*.

INVCDF:
This function computes the quantile from any continuous probability distribution by numerically inverting the CDF of that distribution. You need to specify the CDF function of the distribution, the values of its parameters, and the cumulative probability to compute the quantile.

LIMMOMENT:
This function computes the limited moment of order k with upper limit u for any continuous probability distribution. The limited moment is defined as

$$\begin{aligned} E[(X \wedge u)^k] &= \int_0^u x^k f(x)dx + \int_u^\infty u^k f(x)dx \\ &= \int_0^u x^k f(x)dx + u^k(1 - F(u)) \end{aligned}$$

where $f(x)$ and $F(x)$ denote the PDF and the CDF of the distribution, respectively. The LIMMOMENT function uses the following alternate definition, which can be derived using integration-by-parts:

$$E[(X \wedge u)^k] = k \int_0^u (1 - F(x))x^{k-1} dx$$

You need to specify the CDF function of the distribution, the values of its parameters, and the values of k and u to compute the limited moment.

Custom Objective Functions (Experimental)

You can use a series of programming statements that use variables in the input data set specified by DATA= option in the PROC HPSEVERITY statement to assign a value to an objective function symbol. The objective function symbol must be specified using the OBJECTIVE= option in the PROC HPSEVERITY statement.

The objective function can be programmed such that it is applicable to any distribution that is used in the model. For that purpose, PROC HPSEVERITY recognizes the following *keyword* functions in the programming statements:

PDF(x)	returns the probability density function (PDF) of a distribution evaluated at the current value of a data set variable x.
CDF(x)	returns the cumulative distribution function (CDF) of a distribution evaluated at the current value of a data set variable x.
SDF(x)	returns the survival distribution function (SDF) of a distribution evaluated at the current value of a data set variable x.
LOGPDF(x)	returns the natural logarithm of the PDF of a distribution evaluated at the current value of a data set variable x.
LOGCDF(x)	returns the natural logarithm of the CDF of a distribution evaluated at the current value of a data set variable x.
LOGSDF(x)	returns the natural logarithm of the SDF of a distribution evaluated at the current value of a data set variable x.
EDF(x)	returns the empirical distribution function (EDF) estimate evaluated at the current value of a data set variable x. Internally, PROC HPSEVERITY computes the estimate using the SVRTUTIL_EDF function as described in the section "Predefined Utility Functions" on page 204. The EDF estimate required by the SVRTUTIL_EDF function is computed using the response variable values in the current BY group or in the entire input data set if no BY statement is specified.
EMPLIMMOMENT(k, u)	returns the empirical limited moment of order k evaluated at the current value of a data set variable u that represents the upper limit of the limited moment. The order k can also be a data set variable. Internally, PROC HPSEVERITY computes the moment using the SVRTUTIL_EMPLIMMOMENT function as described in the section "Predefined Utility Functions" on page 204. The EDF estimate required by the SVRTUTIL_EMPLIMMOMENT function is computed using the response variable values in the current BY group or in the entire input data set if no BY statement is specified.

LIMMOMENT(k, u)
: returns the limited moment of order k evaluated at the current value of a data set variable u that represents the upper limit of the limited moment. The order k can be a data set variable or a constant. Internally, for each candidate distribution, PROC HPSEVERITY computes the moment using the LIMMOMENT function as described in the section "Predefined Utility Functions" on page 204.

All the preceding functions are right-hand side functions. They act as placeholders for distribution-specific functions, with the exception of _EDF_ and _EMPLIMMOMENT_ functions. As an example, let the data set Work.Test contain a response variable Y and a left-truncation threshold variable T. The following statements use the values in this data set to fit a model with distribution D such that the parameters of the model minimize the value of the objective function symbol MYOBJ:

```
options cmplib=(work.mydist);
proc hpseverity data=work.test objective=myobj;
    loss y / lt=t;

    myobj = -_LOGPDF_(y);
    if (not(missing(t))) then
        myobj = myobj + log(1-_CDF_(t));

    dist d;
run;
```

The symbol MYOBJ is designated as an objective function symbol by using the OBJECTIVE= option in the PROC HPSEVERITY statement. The response variable Y and left-truncation variable T are specified in the LOSS statement. The distribution D is specified in the DIST statement. The remaining statements constitute a program that computes the value of the MYOBJ symbol.

Let the distribution D have parameters P1 and P2. In order to estimate the model for this distribution, PROC HPSEVERITY internally converts the generic program to the following program specific to distribution D:

```
myobj = -D_LOGPDF(y, p1, p2);
if (not(missing(t))) then
    myobj = myobj + log(1-D_CDF(t, p1, p2));
```

Note that the generic keyword functions _LOGPDF_ and _CDF_ have been replaced with distribution-specific functions D_LOGPDF and D_CDF, respectively, with appropriate distribution parameters. The D_LOGPDF and D_CDF functions must have been defined previously and are assumed to be available in the Work.Mydist library specified in the CMPLIB= option.

The program is executed for each observation in Work.Test to compute the value of MYOBJ by using the values of variables Y and T in that observation and internally computed values of the model parameters P1 and P2. The values of MYOBJ are then added over all the observations of the data set or over all the observations of the current BY group if a BY statement is specified. The resulting aggregate value is the value of the objective function, and it is supplied to the optimizer. If the optimizer requires derivatives of the objective function, then PROC HPSEVERITY automatically differentiates MYOBJ with respect to the parameters P1 and P2. The optimizer iterates over various combinations of the values of parameters P1 and P2, each time computing a new value of the objective function and the needed derivatives of it, until it finds a combination that minimizes the objective function.

Note the following points when you define your own program to compute the custom objective function:

- The value of the objective function is always minimized by PROC HPSEVERITY. If you want to maximize the value of a certain objective, then add a statement that assigns the negated value of the maximization objective to the objective function symbol specified in the OBJECTIVE= option. Minimization of the negated objective is equivalent to the maximization of the original objective.

- The contributions of individual observations are always added to compute the overall objective function in a given iteration of the optimizer. If you have specified the WEIGHT statement, then the contribution of each observation is weighted by multiplying it with the normalized value of the weight variable for that observation.

- If you are fitting multiple distributions in one PROC HPSEVERITY step and use any of the keyword functions in your program, then it is recommended that you do not explicitly use the parameters of any of the specified distributions in your programming statements.

- If you use a specific keyword function in your programming statements, then the corresponding distribution functions must be defined in a library specified in the CMPLIB= system option or in Sashelp.Svrtdist, the predefined functions library. In the preceding example, it is assumed that the functions D_LOGPDF and D_CDF are defined in the Work.Mydist library specified in the CMPLIB= option.

- You can use most DATA step statements and functions in your program. The DATA step file and the data set I/O statements (for example, INPUT, FILE, SET, and MERGE) are not available. However, some functionality of the PUT statement is supported. See the section "PROC FCMP and DATA Step Differences" in *Base SAS Procedures Guide* for more information. In addition to the differences listed in that section, the following differences exist:

  - Only numeric-valued variables can be used in PROC HPSEVERITY programming statements. This restriction also implies that you cannot use SAS functions or call routines that require character-valued arguments, unless you pass those arguments as constant (literal) strings or characters.
  - You cannot use functions that create lagged versions of a variable in PROC HPSEVERITY programming statements. If you need lagged versions, then you can use a DATA step prior to the PROC HPSEVERITY step to add those versions to the input data set.

- When coding your programming statements, avoid defining variables that begin with an underscore (_), because they might conflict with internal variables created by PROC HPSEVERITY.

Custom Objective Functions and Regression Effects

If you have specified regressors using the SCALEMODEL statement, then PROC HPSEVERITY automatically adds a statement prior to your programming statements to compute the value of the scale parameter or the log-transformed scale parameter of the distribution using the values of the regression variables and internally created regression parameters. For example, if you have specified three regressors x1, x2, and x3 in the SCALEMODEL statement, then for a model that contains the distribution D with scale parameter S, PROC HPSEVERITY prepends your program with a statement that is equivalent to the following statement:

```
S = _SEVTHETA0 * exp(_SEVBETA1 * x1 + _SEVBETA2 * x2 + _SEVBETA3 * x3);
```

If a model contains a distribution D1 with a log-transformed scale parameter M, PROC HPSEVERITY prepends your program with a statement that is equivalent to the following statement:

```
M = _SEVTHETA0 + _SEVBETA1 * x1 + _SEVBETA2 * x2 + _SEVBETA3 * x3;
```

The _SEVTHETA0, _SEVBETA1, _SEVBETA2, and _SEVBETA3 are the internal regression parameters associated with the intercept and the regressors x1, x2, and x3, respectively.

Since the names of the internal regression parameters start with a prefix _SEV, if you use a variable in your program with a name that begins with _SEV, then PROC HPSEVERITY writes an error message to the SAS log and stops processing.

Input Data Sets

PROC HPSEVERITY accepts DATA= and INEST= data sets as input data sets. This section details the information they are expected to contain.

DATA= Data Set

The DATA= data set is expected to contain the values of the analysis variables specified in the LOSS statement and the SCALEMODEL statement.

INEST= Data Set

The INEST= data set is expected to contain the initial values of the parameters for the parameter estimation process. The data set must contain the following variables:

MODEL identifying name of the distribution for which the estimates are provided.

TYPE type of the estimate. The value of this variable must be EST for an observation to be valid.

<Parameter 1> . . . <Parameter M>
M variables, named after the parameters of all candidate distributions, that contain initial values of the respective parameters. M is the cardinality of the union of parameter name sets from all candidate distributions. In an observation, estimates are read only from variables for parameters that correspond to the distribution specified by the _MODEL_ variable.

If you specify a missing value for some parameters, then default initial values are used unless the parameter is initialized by using the INIT= option in the DIST statement. If you want to use the *dist*_PARMINIT subroutine for initializing the parameters of a model, then you should either not specify the model in the INEST= data set or specify missing values for all the distribution parameters in the INEST= data set and not use the INIT= option in the DIST statement.

If regressors are specified, then the initial value provided for the first parameter of each distribution must be the base value of the scale or log-transformed scale parameter. For more information, see the section "Estimating Regression Effects" on page 175.

<Regressor 1> . . . <Regressor K>
If K regressors are specified in the SCALEMODEL statement, then the INEST= data set must contain K variables that are named for each regressor. The variables contain initial values of the respective regression coefficients. If a regressor is linearly dependent on other regressors, then you can indicate this by providing a special missing value of .R for the respective variable. If a variable is marked as linearly dependent for one model, then it must be marked so for all the models. Similarly, if a variable is not marked as linearly dependent for one model, then it must be marked so for all the models.

Output Data Sets

PROC HPSEVERITY writes the OUTEST=, OUTMODELINFO=, and OUTSTAT= data sets when requested by their respective options. The data sets and their contents are described in the following sections.

OUTEST= Data Set

The OUTEST= data set records the estimates of the model parameters. It also contains estimates of their standard errors and optionally their covariance structure.

If the COVOUT option is not specified, then the data set contains the following variables:

MODEL identifying name of the distribution model. The observation contains information about this distribution.

TYPE type of the estimates reported in this observation. It can take one of the following two values:

EST	point estimates of model parameters
STDERR	standard error estimates of model parameters

STATUS status of the reported estimates. The possible values are listed in the section "_STATUS_ Variable Values" on page 215.

<Parameter 1> . . . <Parameter M>
M variables, named after the parameters of all candidate distributions, that contain estimates of the respective parameters. M is the cardinality of the union of parameter name sets from all candidate distributions. In an observation, estimates are populated only for parameters that correspond to the distribution specified by the _MODEL_ variable. If _TYPE_ is EST, then the estimates are missing if the model does not converge. If _TYPE_ is STDERR, then the estimates are missing if covariance estimates cannot be obtained.

If regressors are specified, then the estimate reported for the first parameter of each distribution is the estimate of the base value of the scale or log-transformed scale parameter. For more information, see the section "Estimating Regression Effects" on page 175.

<Regressor 1> . . . <Regressor K>
If K regressors are specified in the SCALEMODEL statement, then the OUTEST= data set contains K variables that are named for each regressor. The variables contain

estimates for their respective regression coefficients. If a regressor is deemed to be linearly dependent on other regressors, then a warning message is printed to the SAS log and a special missing value of .R is written in the respective variable. If _TYPE_ is EST, then the estimates are missing if the model does not converge. If _TYPE_ is STDERR, then the estimates are missing if covariance estimates cannot be obtained.

If the COVOUT option is specified, then the OUTEST= data set contains additional observations that contain the estimates of the covariance structure. Given the symmetric nature of the covariance structure, only the lower triangular portion is reported. In addition to the variables listed and described previously, the data set contains the following variables that are either new or have a modified description:

TYPE	type of the estimates reported in this observation. For observations that contain rows of the covariance structure, the value is COV.
STATUS	status of the reported estimates. For observations that contain rows of the covariance structure, the status is 0 if covariance estimation was successful. If estimation fails, the status is 1 and a single observation is reported with _TYPE_=COV and missing values for all the parameter variables.
NAME	Name of the parameter for the row of covariance matrix reported in the current observation.

OUTMODELINFO= Data Set

The OUTMODELINFO= data set records the information about each specified distribution. It contains the following variables:

MODEL	identifying name of the distribution model. The observation contains information about this distribution.
DESCRIPTION	descriptive name of the model. This has a nonmissing value only if the DESCRIPTION function has been defined for this model.
VALID	validity of the model. This has a value of 1 for valid models and a value of 0 for invalid models.
_PARMNAME1 ... _PARMNAMEM	M variables that contain names of parameters of the distribution model, where M is the maximum number of parameters across all the specified distribution models. For a given distribution with m parameters, values of variables _PARMNAMEj $(j > m)$ are missing.

OUTSTAT= Data Set

The OUTSTAT= data set records statistics of fit and model selection information. The data set contains the following variables:

MODEL	identifying name of the distribution model. The observation contains information about this distribution.
NMODELPARM	number of parameters in the distribution.
NESTPARM	number of estimated parameters. This includes the regression parameters, if any regressors are specified.

NOBS	number of nonmissing observations used for parameter estimation.
STATUS	status of the parameter estimation process for this model. The possible values are listed in the section "_STATUS_ Variable Values" on page 215.
SELECTED	indicator of the best distribution model. If the value is 1, then this model is the best model according to the specified model selection criterion. This value is missing if the parameter estimation process does not converge for this model.
Neg2LogLike	value of the log likelihood, multiplied by –2, that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.
AIC	value of the Akaike's information criterion (AIC) that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.
AICC	value of the corrected Akaike's information criterion (AICC) that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.
BIC	value of the Schwarz Bayesian information criterion (BIC) that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.
KS	value of the Kolmogorov-Smirnov (KS) statistic that is attained at the end of the parameter estimation process. This value is missing if parameter estimation process does not converge for this model.
AD	value of the Anderson-Darling (AD) statistic that is attained at the end of the parameter estimation process. This value is missing if parameter estimation process does not converge for this model.
CVM	value of the Cramér-von Mises (CvM) statistic that is attained at the end of the parameter estimation process. This value is missing if parameter estimation process does not converge for this model.

STATUS Variable Values

The _STATUS_ variable in the OUTEST= and OUTSTAT= data sets contains a value that indicates the status of the parameter estimation process for the respective distribution model. The variable can take the following values in the OUTEST= data set for _TYPE_=EST observations and in the OUTSTAT= data set:

0	The parameter estimation process converged for this model.
301	The parameter estimation process might not have converged for this model because there is no improvement in the objective function value. This might indicate that the initial values of the parameters are optimal, or you can try different convergence criteria in the NLOPTIONS statement.
302	The parameter estimation process might not have converged for this model because the number of iterations exceeded the maximum allowed value. You can try setting a larger value for the MAXITER= options in the NLOPTIONS statement.
303	The parameter estimation process might not have converged for this model because the number of objective function evaluations exceeded the maximum allowed value. You can try setting a larger value for the MAXFUNC= options in the NLOPTIONS statement.

304	The parameter estimation process might not have converged for this model because the time taken by the process exceeded the maximum allowed value. You can try setting a larger value for the MAXTIME= option in the NLOPTIONS statement.
400	The parameter estimation process did not converge for this model.

The _STATUS_ variable can take the following values in the OUTEST= data set for _TYPE_=STDERR and _TYPE_=COV observations:

0	The covariance and standard error estimates are available and valid.
1	The covariance and standard error estimates are not available, because the process of computing covariance estimates failed.

Displayed Output

The HPSEVERITY procedure optionally produces displayed output by using the Output Delivery System (ODS). All output is controlled by the PRINT= option in the PROC HPSEVERITY statement. Table 5.7 relates the ODS tables to PRINT= options.

Table 5.7 ODS Tables Produced in PROC HPSEVERITY

ODS Table Name	Description	Option
AllFitStatistics	Statistics of fit for all the distribution models	PRINT=ALLFITSTATS
ConvergenceStatus	Convergence status of parameter estimation process	PRINT=CONVSTATUS
DescStats	Descriptive statistics for the response variable	PRINT=DESCSTATS
DistributionInfo	Distribution information	PRINT=DISTINFO
EstimationDetails	Details of the estimation process for all the distribution models	PRINT=ESTIMATIONDETAILS
InitialValues	Initial parameter values and bounds	PRINT=INITIALVALUES
IterationHistory	Optimization iteration history	PRINT=NLOHISTORY
ModelSelection	Model selection summary	PRINT=SELECTION
OptimizationSummary	Optimization summary	PRINT=NLOSUMMARY
ParameterEstimates	Final parameter estimates	PRINT=ESTIMATES
PerformanceInfo	Execution environment information that pertains to the computational performance	Default
RegDescStats	Descriptive statistics for the regressor variables	PRINT=DESCSTATS
StatisticsOfFit	Statistics of fit	PRINT=STATISTICS

Table 5.7 *continued*

ODS Table Name	Description	Option
Timing	Timing information for various computational stages of the procedure	DETAILS (PERFORMANCE statement)

PRINT= Option

If the PRINT= option is not specified, then by default PROC HPSEVERITY produces the ModelSelection, PerformanceInfo, ConvergenceStatus, OptimizationSummary, StatisticsOfFit, and ParameterEstimates ODS tables.

You can specify the following values for the PRINT= option:

PRINT=ALLFITSTATS
displays the comparison of all the statistics of fit for all the models in one table. The table does not include the models whose parameter estimation process does not converge. If all the models fail to converge, then this table is not produced. If the table contains more than one model, then the best model according to each statistic is indicated with an asterisk (*) in that statistic's column.

PRINT=CONVSTATUS
displays the convergence status of the parameter estimation process.

PRINT=DESCSTATS
displays the descriptive statistics for the response variable. If regressor variables are specified, a table with their descriptive statistics is also displayed.

PRINT=DISTINFO
displays the information about all the candidate distribution. It includes the name, the description, the number of distribution parameters, and whether the distribution is valid for the specified modeling task.

PRINT=ESTIMATES
displays the final estimates of parameters. The estimates are not displayed for models whose parameter estimation process does not converge.

PRINT=ESTIMATIONDETAILS
displays the comparative details of the estimation process that is used to fit each candidate distribution. If you specify the DETAILS option in the PERFORMANCE statement, then this table contains a column that indicates the time taken to estimate each candidate model.

PRINT=INITIALVALUES
displays the initial values and bounds used for estimating each model.

PRINT=NLOHISTORY
displays the iteration history of the nonlinear optimization process used for estimating the parameters.

PRINT=NLOSUMMARY
displays the summary of the nonlinear optimization process used for estimating the parameters.

PRINT=SELECTION
displays the model selection table. The table shows the convergence status of each candidate model, and the value of the selection criterion along with an indication of the selected model.

PRINT=STATISTICS
displays the statistics of fit for each model. The statistics of fit are not displayed for models whose parameter estimation process does not converge.

Performance Information

The “Performance Information” table is produced by default. It displays information about the execution mode. For single-machine mode, the table displays the number of threads used. For distributed mode, the table displays the grid mode (symmetric or asymmetric), the number of compute nodes, and the number of threads per node.

If you specify the DETAILS option in the PERFORMANCE statement, the procedure also produces a “Timing” table in which elapsed times (absolute and relative) for the main tasks of the procedure are displayed.

Examples: HPSEVERITY Procedure

Example 5.1: Defining a Model for Gaussian Distribution

Suppose you want to fit a distribution model other than one of the predefined ones available to you. Suppose you want to define a model for the Gaussian distribution with the following typical parameterization of the PDF (f) and CDF (F):

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$
$$F(x;\mu,\sigma) = \frac{1}{2}\left(1+\operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right)$$

For PROC HPSEVERITY, a *distribution model* consists of a set of functions and subroutines that are defined with the FCMP procedure. Each function and subroutine should be written following certain rules. For more information, see the section “Defining a Severity Distribution Model with the FCMP Procedure” on page 192.

NOTE: The Gaussian distribution is not a commonly used severity distribution. It is used in this example primarily to illustrate the process of defining your own distribution models. Although the distribution has a support over the entire real line, you can fit the distribution with PROC HPSEVERITY only if the input sample contains nonnegative values.

The following SAS statements define a distribution model named NORMAL for the Gaussian distribution. The OUTLIB= option in the PROC FCMP statement stores the compiled versions of the functions and subroutines in the ‘models’ package of the Work.Sevexmpl library. The LIBRARY= option in the PROC

FCMP statement enables this PROC FCMP step to use the SVRTUTIL_RAWMOMENTS utility subroutine that is available in the Sashelp.Svrtdist library. The subroutine is described in the section "Predefined Utility Functions" on page 204.

```
/*-------- Define Normal Distribution with PROC FCMP  ----------*/
proc fcmp library=sashelp.svrtdist outlib=work.sevexmpl.models;
   function normal_pdf(x,Mu,Sigma);
      /* Mu    : Location */
      /* Sigma : Standard Deviation */
      return ( exp(-(x-Mu)**2/(2 * Sigma**2)) /
             (Sigma * sqrt(2*constant('PI'))) );
   endsub;

   function normal_cdf(x,Mu,Sigma);
      /* Mu    : Location */
      /* Sigma : Standard Deviation */
      z = (x-Mu)/Sigma;
      return (0.5 + 0.5*erf(z/sqrt(2)));
   endsub;

   subroutine normal_parminit(dim, x[*], nx[*], F[*], Ftype, Mu, Sigma);
      outargs Mu, Sigma;
      array m[2] / nosymbols;

      /* Compute estimates by using method of moments */
      call svrtutil_rawmoments(dim, x, nx, 2, m);
      Mu    = m[1];
      Sigma = sqrt(m[2] - m[1]**2);
   endsub;

   subroutine normal_lowerbounds(Mu, Sigma);
      outargs Mu, Sigma;
      Mu = .;    /* Mu has no lower bound */
      Sigma = 0; /* Sigma > 0 */
   endsub;
quit;
```

The statements define the two functions required of any distribution model (NORMAL_PDF and NORMAL_CDF) and two optional subroutines (NORMAL_PARMINIT and NORMAL_LOWERBOUNDS). The name of each function or subroutine must follow a specific structure. It should start with the model's short or identifying name, which is 'NORMAL' in this case, followed by an underscore '_', followed by a keyword suffix such as 'PDF'. Each function or subroutine has a specific purpose. For more information about all the functions and subroutines that you can define for a distribution model, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 192. Following is the description of each function and subroutine defined in this example:

- The PDF and CDF suffixes define functions that return the probability density function and cumulative distribution function values, respectively, given the values of the random variable and the distribution parameters.
- The PARMINIT suffix defines a subroutine that returns the initial values for the parameters by using the sample data or the empirical distribution function (EDF) estimate computed from it. In this example, the parameters are initialized by using the method of moments. Hence, you do not need to use the EDF

estimates, which are available in the F array. The first two raw moments of the Gaussian distribution are as follows:

$$E[x] = \mu, \; E[x^2] = \mu^2 + \sigma^2$$

Given the sample estimates, m_1 and m_2, of these two raw moments, you can solve the equations $E[x] = m_1$ and $E[x^2] = m_2$ to get the following estimates for the parameters: $\hat{\mu} = m_1$ and $\hat{\sigma} = \sqrt{m_2 - m_1^2}$. The NORMAL_PARMINIT subroutine implements this solution. It uses the SVRTUTIL_RAWMOMENTS utility subroutine to compute the first two raw moments.

- The LOWERBOUNDS suffix defines a subroutine that returns the lower bounds on the parameters. PROC HPSEVERITY assumes a default lower bound of 0 for all the parameters when a LOWERBOUNDS subroutine is not defined. For the parameter μ (*Mu*), there is no lower bound, so you need to define the NORMAL_LOWERBOUNDS subroutine. It is recommended that you assign bounds for all the parameters when you define the LOWERBOUNDS subroutine or its counterpart, the UPPERBOUNDS subroutine. Any unassigned value is returned as a missing value, which PROC HPSEVERITY interprets to mean that the parameter is unbounded, and that might not be what you want.

You can now use this distribution model with PROC HPSEVERITY. Let the following DATA step statements simulate a normal sample with $\mu = 10$ and $\sigma = 2.5$:

```
/*-------- Simulate a Normal sample ----------*/
data testnorm(keep=y);
   call streaminit(12345);
   do i=1 to 100;
      y = rand('NORMAL', 10, 2.5);
      output;
   end;
run;
```

Prior to using your distribution with PROC HPSEVERITY, you must communicate the location of the library that contains the definition of the distribution and the locations of libraries that contain any functions and subroutines used by your distribution model. The following OPTIONS statement sets the CMPLIB= system option to include the FCMP library Work.Sevexmpl in the search path used by PROC HPSEVERITY to find FCMP functions and subroutines.

```
/*--- Set the search path for functions defined with PROC FCMP ---*/
options cmplib=(work.sevexmpl);
```

Now, you are ready to fit the NORMAL distribution model with PROC HPSEVERITY. The following statements fit the model to the values of Y in the Work.Testnorm data set:

```
/*--- Fit models with PROC HPSEVERITY ---*/
proc hpseverity data=testnorm print=all;
   loss y;
   dist Normal;
run;
```

The DIST statement specifies the identifying name of the distribution model, which is 'NORMAL'. Neither is the INEST= option specified in the PROC HPSEVERITY statement nor is the INIT= option specified in the DIST statement. So, PROC HPSEVERITY initializes the parameters by invoking the NORMAL_PARMINIT subroutine.

Some of the results prepared by the preceding PROC HPSEVERITY step are shown in Output 5.1.1 and Output 5.1.2. The descriptive statistics of variable Y and the "Model Selection" table, which includes just the normal distribution, are shown in Output 5.1.1.

Output 5.1.1 Summary of Results for Fitting the Normal Distribution

```
                     The HPSEVERITY Procedure

                         Input Data Set

                      Name    WORK.TESTNORM


                  Descriptive Statistics for y

         Observations                                 100
         Observations Used for Estimation             100
         Minimum                                  3.88249
         Maximum                                 16.00864
         Mean                                    10.02059
         Standard Deviation                       2.37730


                         Model Selection

                                      -2 Log
      Distribution    Converged    Likelihood    Selected

      Normal          Yes           455.97541    Yes
```

The initial values for the parameters, the optimization summary, and the final parameter estimates are shown in Output 5.1.2. No iterations are required to arrive at the final parameter estimates, which are identical to the initial values. This confirms the fact that the maximum likelihood estimates for the Gaussian distribution are identical to the estimates obtained by the method of moments that was used to initialize the parameters in the NORMAL_PARMINIT subroutine.

Output 5.1.2 Details of the Fitted Normal Distribution Model

```
                     The HPSEVERITY Procedure
                       Normal Distribution

                    Distribution Information

               Name                          Normal
               Distribution Parameters            2


              Initial Parameter Values and Bounds

                      Initial          Lower         Upper
      Parameter         Value          Bound         Bound

      Mu             10.02059         -Infty         Infty
      Sigma           2.36538    1.05367E-8         Infty
```

Output 5.1.2 *continued*

```
                         Optimization Summary

              Optimization Technique    Trust Region
              Iterations                           0
              Function Calls                       4
              Log Likelihood              -227.98770


                         Parameter Estimates

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

   Mu              10.02059         0.23894       41.94         <.0001
   Sigma            2.36538         0.16896       14.00         <.0001
```

The NORMAL distribution defined and illustrated here has no scale parameter, because all the following inequalities are true:

$$
\begin{aligned}
f(x;\mu,\sigma) &\neq \frac{1}{\mu} f(\frac{x}{\mu};1,\sigma) \\
f(x;\mu,\sigma) &\neq \frac{1}{\sigma} f(\frac{x}{\sigma};\mu,1) \\
F(x;\mu,\sigma) &\neq F(\frac{x}{\mu};1,\sigma) \\
F(x;\mu,\sigma) &\neq F(\frac{x}{\sigma};\mu,1)
\end{aligned}
$$

This implies that you cannot estimate the effect of regressors on a model for the response variable based on this distribution.

Example 5.2: Defining a Model for the Gaussian Distribution with a Scale Parameter

If you want to estimate the effects of regressors, then the model needs to be parameterized to have a scale parameter. Although this might not be always possible, it is possible for the Gaussian distribution by replacing the location parameter μ with another parameter, $\alpha = \mu/\sigma$, and defining the PDF (f) and the CDF (F) as follows:

$$
\begin{aligned}
f(x;\sigma,\alpha) &= \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x}{\sigma}-\alpha\right)^2\right) \\
F(x;\sigma,\alpha) &= \frac{1}{2}\left(1+\operatorname{erf}\left(\frac{1}{\sqrt{2}}\left(\frac{x}{\sigma}-\alpha\right)\right)\right)
\end{aligned}
$$

You can verify that σ is the scale parameter, because both of the following equalities are true:

$$f(x;\sigma,\alpha) = \frac{1}{\sigma} f(\frac{x}{\sigma};1,\alpha)$$
$$F(x;\sigma,\alpha) = F(\frac{x}{\sigma};1,\alpha)$$

NOTE: The Gaussian distribution is not a commonly used severity distribution. It is used in this example primarily to illustrate the concept of parameterizing a distribution such that it has a scale parameter. Although the distribution has a support over the entire real line, you can fit the distribution with PROC HPSEVERITY only if the input sample contains nonnegative values.

The following statements use the alternate parameterization to define a new model named NORMAL_S. The definition is stored in the Work.Sevexmpl library.

```
/*-------- Define Normal Distribution With Scale Parameter  ----------*/
proc fcmp library=sashelp.svrtdist outlib=work.sevexmpl.models;
   function normal_s_pdf(x, Sigma, Alpha);
      /* Sigma : Scale & Standard Deviation */
      /* Alpha : Scaled mean */
      return ( exp(-(x/Sigma - Alpha)**2/2) /
               (Sigma * sqrt(2*constant('PI'))) );
   endsub;

   function normal_s_cdf(x, Sigma, Alpha);
      /* Sigma : Scale & Standard Deviation */
      /* Alpha : Scaled mean */
      z = x/Sigma - Alpha;
      return (0.5 + 0.5*erf(z/sqrt(2)));
   endsub;

   subroutine normal_s_parminit(dim, x[*], nx[*], F[*], Ftype, Sigma, Alpha);
      outargs Sigma, Alpha;
      array m[2] / nosymbols;

      /* Compute estimates by using method of moments */
      call svrtutil_rawmoments(dim, x, nx, 2, m);
      Sigma = sqrt(m[2] - m[1]**2);
      Alpha = m[1]/Sigma;
   endsub;

   subroutine normal_s_lowerbounds(Sigma, Alpha);
      outargs Sigma, Alpha;
      Alpha = .; /* Alpha has no lower bound */
      Sigma = 0; /* Sigma > 0 */
   endsub;
quit;
```

An important point to note is that the scale parameter *Sigma* is the first distribution parameter (after the ‘x’ argument) listed in the signatures of NORMAL_S_PDF and NORMAL_S_CDF functions. *Sigma* is also the first distribution parameter listed in the signatures of other subroutines. This is required by PROC HPSEVERITY, so that it can identify which is the scale parameter. When regressor variables are specified, PROC HPSEVERITY checks whether the first parameter of each candidate distribution is a scale parameter (or a log-transformed scale parameter if *dist*_SCALETRANSFORM subroutine is defined for the distribution with LOG as the transform). If it is not, then an appropriate message is written the SAS log and that distribution is not fitted.

Let the following DATA step statements simulate a sample from the normal distribution where the parameter σ is affected by the regressors as follows:

$$\sigma = \exp(1 + 0.5\ \text{X1} + 0.75\ \text{X3} - 2\ \text{X4} + \text{X5})$$

The sample is simulated such that the regressor X2 is linearly dependent on regressors X1 and X3.

```
/*--- Simulate a Normal sample affected by Regressors ---*/
data testnorm_reg(keep=y x1-x5 Sigma);
   array x{*} x1-x5;
   array b{6} _TEMPORARY_ (1 0.5 . 0.75 -2 1);
   call streaminit(34567);
   label y='Normal Response Influenced by Regressors';

   do n = 1 to 100;
      /* simulate regressors */
      do i = 1 to dim(x);
         x(i) = rand('UNIFORM');
      end;
      /* make x2 linearly dependent on x1 */
      x(2) = 5 * x(1);

      /* compute log of the scale parameter */
      logSigma = b(1);
      do i = 1 to dim(x);
         if (i ne 2) then
            logSigma = logSigma + b(i+1) * x(i);
      end;

      Sigma = exp(logSigma);
      y = rand('NORMAL', 25, Sigma);

      output;
   end;
run;
```

The following statements use PROC HPSEVERITY to fit the NORMAL_S distribution model along with some of the predefined distributions to the simulated sample:

```
/*--- Set the search path for functions defined with PROC FCMP ---*/
options cmplib=(work.sevexmpl);

/*-------- Fit models with PROC HPSEVERITY --------*/
proc hpseverity data=testnorm_reg print=all;
   loss y;
   scalemodel x1-x5;
   dist Normal_s burr logn pareto weibull;
run;
```

The “Model Selection” table in Output 5.2.1 indicates that all the models, except the Burr distribution model, have converged. Also, only three models, Normal_s, Burr, and Weibull, seem to have a good fit for the data. The table that compares all the fit statistics indicates that Normal_s model is the best according to the likelihood-based statistics.

Output 5.2.1 Summary of Results for Fitting the Normal Distribution with Regressors

```
                         The HPSEVERITY Procedure

                              Input Data Set

                         Name    WORK.TESTNORM_REG


                             Model Selection

                                                -2 Log
              Distribution    Converged    Likelihood    Selected

              Normal_s        Yes           603.95786    Yes
              Burr            Maybe         612.81685    No
              Logn            Yes           749.20125    No
              Pareto          Yes           841.07022    No
              Weibull         Yes           612.77496    No


                            All Fit Statistics

                    -2 Log
 Distribution   Likelihood          AIC          AICC          BIC           KS

 Normal_s       603.95786*   615.95786*   616.86108*   631.58888*      1.52388
 Burr           612.81685    626.81685    628.03424    645.05304       1.50448*
 Logn           749.20125    761.20125    762.10448    776.83227       2.88110
 Pareto         841.07022    853.07022    853.97345    868.70124       4.83810
 Weibull        612.77496    624.77496    625.67819    640.40598       1.50490

                  Note: The asterisk (*) marks the best model
                     according to each column's criterion.

                             All Fit Statistics

                    Distribution          AD            CvM

                    Normal_s         4.00152        0.70769
                    Burr             3.90072*       0.63399*
                    Logn            16.20558        3.04825
                    Pareto          31.60568        6.84046
                    Weibull          3.90559        0.63458

                     Note: The asterisk (*) marks the best
                            model according to each
                              column's criterion.
```

This prompts you to further evaluate why the model with Burr distribution has not converged. The initial values, convergence status, and the optimization summary for the Burr distribution are shown in Output 5.2.2. The initial values table indicates that the regressor X2 is redundant, which is expected. More importantly, the convergence status indicates that it requires more than 50 iterations. PROC HPSEVERITY enables you to change several settings of the optimizer by using the NLOPTIONS statement. In this case, you can increase the limit of 50 on the iterations, change the convergence criterion, or change the technique to something other than the default trust-region technique.

Output 5.2.2 Details of the Fitted Burr Distribution Model

```
                   The HPSEVERITY Procedure
                      Burr Distribution

                   Distribution Information

         Name                                      Burr
         Description                  Burr Distribution
         Distribution Parameters                      3
         Regression Parameters                        4


             Initial Parameter Values and Bounds

                      Initial         Lower         Upper
     Parameter          Value         Bound         Bound

     Theta           25.75198    1.05367E-8         Infty
     Alpha            2.00000    1.05367E-8         Infty
     Gamma            2.00000    1.05367E-8         Infty
     x1               0.07345    -709.78271     709.78271
     x2             Redundant
     x3              -0.14056    -709.78271     709.78271
     x4               0.27064    -709.78271     709.78271
     x5              -0.23230    -709.78271     709.78271


                      Convergence Status

                Needs more than 50 iterations.


                     Optimization Summary

           Optimization Technique    Trust Region
           Iterations                          50
           Function Calls                     137
           Log Likelihood              -306.40842
```

The following PROC HPSEVERITY step uses the NLOPTIONS statement to change the convergence criterion and the limits on the iterations and function evaluations, exclude the lognormal and Pareto distributions that have been confirmed previously to fit the data poorly, and exclude the redundant regressor X2 from the model:

```
/*--- Refit and compare models with higher limit on iterations ---*/
proc hpseverity data=testnorm_reg print=all;
   loss y;
   scalemodel x1 x3-x5;
   dist Normal_s burr weibull;
   nloptions absfconv=2.0e-5 maxiter=100 maxfunc=500;
run;
```

The results shown in Output 5.2.3 indicate that the Burr distribution has now converged and that the Burr and Weibull distributions have an almost identical fit for the data. The NORMAL_S distribution is still the best distribution according to the likelihood-based criteria.

Output 5.2.3 Summary of Results after Changing Maximum Number of Iterations

```
                         The HPSEVERITY Procedure

                             Input Data Set

                        Name    WORK.TESTNORM_REG


                             Model Selection

                                              -2 Log
             Distribution    Converged    Likelihood    Selected

             Normal_s        Yes           603.95786    Yes
             Burr            Yes           612.79276    No
             Weibull         Yes           612.77496    No


                           All Fit Statistics

                   -2 Log
Distribution    Likelihood          AIC          AICC           BIC            KS

Normal_s        603.95786*    615.95786*    616.86108*    631.58888*      1.52388
Burr            612.79276     626.79276     628.01015     645.02895       1.50472*
Weibull         612.77496     624.77496     625.67819     640.40598       1.50490

                 Note: The asterisk (*) marks the best model
                    according to each column's criterion.

                           All Fit Statistics

                   Distribution           AD            CvM

                   Normal_s          4.00152        0.70769
                   Burr              3.90351*       0.63433*
                   Weibull           3.90559        0.63458

                    Note: The asterisk (*) marks the best
                           model according to each
                             column's criterion.
```

Example 5.3: Defining a Model for Mixed-Tail Distributions

In some applications, a few severity values tend to be extreme as compared to the typical values. The extreme values represent the worst case scenarios and cannot be discarded as outliers. Instead, their distribution must be modeled to prepare for their occurrences. In such cases, it is often useful to fit one distribution to the non-extreme values and another distribution to the extreme values. The *mixed-tail* distribution mixes two distributions: one for the *body* region, which contains the non-extreme values, and another for the *tail* region, which contains the extreme values. The tail distribution is usually a generalized Pareto distribution (GPD), because it is usually good for modeling the conditional excess severity above a threshold. The body distribution can be any distribution. The following definitions are used in describing a generic formulation of a mixed-tail distribution:

$g(x)$	PDF of the body distribution
$G(x)$	CDF of the body distribution
$h(x)$	PDF of the tail distribution
$H(x)$	CDF of the tail distribution
θ	scale parameter for the body distribution
Ω	set of nonscale parameters for the body distribution
ξ	shape parameter for the GPD tail distribution
x_r	normalized value of the response variable at which the tail starts
p_n	mixing probability

Given these notations, the PDF $f(x)$ and the CDF $F(x)$ of the mixed-tail distribution are defined as

$$f(x) = \begin{cases} \frac{p_n}{G(x_b)} g(x) & \text{if } x \le x_b \\ (1 - p_n) h(x - x_b) & \text{if } x > x_b \end{cases}$$

$$F(x) = \begin{cases} \frac{p_n}{G(x_b)} G(x) & \text{if } x \le x_b \\ p_n + (1 - p_n) H(x - x_b) & \text{if } x > x_b \end{cases}$$

where $x_b = \theta x_r$ is the value of the response variable at which the tail starts.

These definitions indicate the following:

- The body distribution is conditional on $X \le x_b$, where X denotes the random response variable.
- The tail distribution is the generalized Pareto distribution of the $(X - x_b)$ values.
- The probability that a response variable value belongs to the body is p_n. Consequently the probability that the value belongs to the tail is $(1 - p_n)$.

The parameters of this distribution are θ, Ω, ξ, x_r, and p_n. The scale of the GPD tail distribution θ_t is computed as

$$\theta_t = \frac{G(x_b; \theta, \Omega)}{g(x_b; \theta, \Omega)} \frac{(1 - p_n)}{p_n} = \theta \frac{G(x_r; \theta = 1, \Omega)}{g(x_r; \theta = 1, \Omega)} \frac{(1 - p_n)}{p_n}$$

The parameter x_r is usually estimated using a tail index estimation algorithm. One such algorithm is the Hill's algorithm (Danielsson et al. 2001), which is implemented by the predefined utility function SVRTUTIL_HILLCUTOFF available to you in the Sashelp.Svrtdist library. The algorithm and the utility function are described in detail in the section "Predefined Utility Functions" on page 204. The function computes an estimate of x_b, which can be used to compute an estimate of x_r because $x_r = x_b / \hat{\theta}$, where $\hat{\theta}$ is the estimate of the scale parameter of the body distribution.

The parameter p_n is usually determined by the domain expert based on the fraction of losses that are expected to belong to the tail.

The following SAS statements define the LOGNGPD distribution model for a mixed-tail distribution with the lognormal distribution as the body distribution and GPD as the tail distribution:

```
/*------- Define Lognormal Body-GPD Tail Mixed Distribution -------*/
proc fcmp library=sashelp.svrtdist outlib=work.sevexmpl.models;
   function LOGNGPD_DESCRIPTION() $256;
      length desc $256;
      desc1 = "Lognormal Body-GPD Tail Distribution.";
      desc2 = " Mu, Sigma, and Xi are free parameters.";
      desc3 = " Xr and Pn are constant parameters.";
      desc = desc1 || desc2 || desc3;
      return(desc);
   endsub;

   function LOGNGPD_SCALETRANSFORM() $3;
      length xform $3;
      xform = "LOG";
      return (xform);
   endsub;

   subroutine LOGNGPD_CONSTANTPARM(Xr,Pn);
   endsub;

   function LOGNGPD_PDF(x, Mu,Sigma,Xi,Xr,Pn);
      cutoff = exp(Mu) * Xr;
      p = CDF('LOGN',cutoff, Mu, Sigma);
      if (x < cutoff + constant('MACEPS')) then do;
         return ((Pn/p)*PDF('LOGN', x, Mu, Sigma));
      end;
      else do;
         gpd_scale = p*((1-Pn)/Pn)/PDF('LOGN', cutoff, Mu, Sigma);
         h = (1+Xi*(x-cutoff)/gpd_scale)**(-1-(1/Xi))/gpd_scale;
         return ((1-Pn)*h);
      end;
   endsub;

   function LOGNGPD_CDF(x, Mu,Sigma,Xi,Xr,Pn);
      cutoff = exp(Mu) * Xr;
      p = CDF('LOGN',cutoff, Mu, Sigma);
      if (x < cutoff + constant('MACEPS')) then do;
         return ((Pn/p)*CDF('LOGN', x, Mu, Sigma));
      end;
      else do;
         gpd_scale = p*((1-Pn)/Pn)/PDF('LOGN', cutoff, Mu, Sigma);
         H = 1 - (1 + Xi*((x-cutoff)/gpd_scale))**(-1/Xi);
         return (Pn + (1-Pn)*H);
      end;
   endsub;

   subroutine LOGNGPD_PARMINIT(dim,x[*],nx[*],F[*],Ftype,
                       Mu,Sigma,Xi,Xr,Pn);
      outargs Mu,Sigma,Xi,Xr,Pn;
      array xe[1] / nosymbols;
      array nxe[1] / nosymbols;

      eps = constant('MACEPS');
```

```
         Pn = 0.8; /* Set mixing probability */
         _status_ = .;
         call streaminit(56789);
         Xb = svrtutil_hillcutoff(dim, x, 100, 25, _status_);
         if (missing(_status_) or _status_ = 1) then
            Xb = svrtutil_percentile(Pn, dim, x, F, Ftype);

         /* prepare arrays for excess values */
         i = 1;
         do while (i <= dim and x[i] < Xb+eps);
            i = i + 1;
         end;
         dime = dim-i+1;
         call dynamic_array(xe, dime);
         call dynamic_array(nxe, dime);
         j = 1;
         do while(i <= dim);
            xe[j] = x[i] - Xb;
            nxe[j] = nx[i];
            i = i + 1;
            j = j + 1;
         end;

         /* Initialize lognormal parameters */
         call logn_parminit(dim, x, nx, F, Ftype, Mu, Sigma);
         if (not(missing(Mu))) then
            Xr = Xb/exp(Mu);
         else
            Xr = .;

         /* Initialize GPD's shape parameter using excess values */
         call gpd_parminit(dime, xe, nxe, F, Ftype, theta_gpd, Xi);
      endsub;

      subroutine LOGNGPD_LOWERBOUNDS(Mu,Sigma,Xi,Xr,Pn);
         outargs Mu,Sigma,Xi,Xr,Pn;

         Mu    = .; /* Mu has no lower bound */
         Sigma = 0; /* Sigma > 0 */
         Xi    = 0; /* Xi > 0 */
      endsub;
   quit;
```

Note the following points about the LOGNGPD definition:

- The parameters x_r and p_n are not estimated with the maximum likelihood method used by PROC HPSEVERITY, so you need to specify them as *constant* parameters by defining the *dist*_CONSTANTPARM subroutine. The signature of LOGNGPD_CONSTANTPARM subroutine lists only the constant parameters *Xr* and *Pn*.

- The parameter x_r is estimated by first using the SVRTUTIL_HILLCUTOFF utility function to compute an estimate of the cutoff point $\hat{x}_b$ and then computing $x_r = \hat{x}_b / e^{\hat{\mu}}$. If SVRTUTIL_HILLCUTOFF fails to compute a valid estimate, then the SVRTUTIL_PERCENTILE utility function is used to set $\hat{x}_b$ to the p_nth percentile of the data. The parameter p_n is fixed to 0.8.

- The Sashelp.Svrtdist library is specified with the LIBRARY= option in the PROC FCMP statement to enable the LOGNGPD_PARMINIT subroutine to use the predefined utility functions (SVRTUTIL_HILLCUTOFF and SVRTUTIL_PERCENTILE) and parameter initialization subroutines (LOGN_PARMINIT and GPD_PARMINIT).
- The LOGNGPD_LOWERBOUNDS subroutine defines the lower bounds for all parameters. This subroutine is required because the parameter *Mu* has a non-default lower bound. The bounds for *Sigma* and *Xi* must be specified. If they are not specified, they are returned as missing values, which PROC HPSEVERITY interprets as having no lower bound. You need not specify any bounds for the constant parameters *Xr* and *Pn*, because they are not subject to optimization.

The following DATA step statements simulate a sample from a mixed-tail distribution with a lognormal body and GPD tail. The parameter p_n is fixed to 0.8, the same value used in the LOGNGPD_PARMINIT subroutine defined previously.

```
/*----- Simulate a sample for the mixed-tail distribution -----*/
data testmixdist(keep=y label='Lognormal Body-GPD Tail Sample');
   call streaminit(45678);
   label y='Response Variable';
   N = 100;
   Mu = 1.5;
   Sigma = 0.25;
   Xi = 1.5;
   Pn = 0.8;

   /* Generate data comprising the lognormal body */
   Nbody = N*Pn;
   do i=1 to Nbody;
      y = exp(Mu) * rand('LOGNORMAL')**Sigma;
      output;
   end;

   /* Generate data comprising the GPD tail */
   cutoff = quantile('LOGNORMAL', Pn, Mu, Sigma);
   gpd_scale = (1-Pn) / pdf('LOGNORMAL', cutoff, Mu, Sigma);
   do i=Nbody+1 to N;
      y = cutoff + ((1-rand('UNIFORM'))**(-Xi) - 1)*gpd_scale/Xi;
      output;
   end;
run;
```

The following statements use PROC HPSEVERITY to fit the LOGNGPD distribution model to the simulated sample. They also fit three other predefined distributions (BURR, LOGN, and GPD). The final parameter estimates are written to the Work.Parmest data set.

```
/*--- Set the search path for functions defined with PROC FCMP ---*/
options cmplib=(work.sevexmpl);

/*-------- Fit LOGNGPD model with PROC HPSEVERITY --------*/
proc hpseverity data=testmixdist print=all outest=parmest;
```

```
      loss y;
      dist logngpd burr logn gpd;
   run;
```

Some of the results prepared by PROC HPSEVERITY are shown in Output 5.3.1 and Output 5.3.2. The "Model Selection" table in Output 5.3.1 indicates that all models converged. The last table in Output 5.3.1 shows that the model with LOGNGPD distribution has the best fit according to almost all the statistics of fit. The Burr distribution model is the closest contender to the LOGNGPD model, but the GPD distribution model fits the data very poorly.

Output 5.3.1 Summary of Fitting Mixed-Tail Distribution

```
                         The HPSEVERITY Procedure

                             Input Data Set

                     Name                  WORK.TESTMIXDIST
                     Label    Lognormal Body-GPD Tail Sample


                             Model Selection

                                            -2 Log
               Distribution    Converged    Likelihood    Selected

               logngpd         Yes           418.78232    Yes
               Burr            Yes           424.93728    No
               Logn            Yes           459.43471    No
               Gpd             Yes           558.13444    No


                           All Fit Statistics

                     -2 Log
Distribution    Likelihood          AIC          AICC           BIC            KS

logngpd         418.78232*   428.78232*   429.42062*   441.80817      0.62140*
Burr            424.93728    430.93728    431.18728    438.75280*     0.71373
Logn            459.43471    463.43471    463.55842    468.64505      1.55267
Gpd             558.13444    562.13444    562.25815    567.34478      3.43470

                   Note: The asterisk (*) marks the best model
                      according to each column's criterion.

                            All Fit Statistics

                     Distribution          AD            CvM

                     logngpd          0.31670*      0.04972*
                     Burr             0.57649       0.07860
                     Logn             3.27122       0.48448
                     Gpd             16.74156       3.31860

                      Note: The asterisk (*) marks the best
                             model according to each
                               column's criterion.
```

The detailed results for the LOGNGPD distribution are shown in Output 5.3.2. The initial values table indicates the values computed by LOGNGPD_PARMINIT subroutine for the *Xr* and *Pn* parameters. It also uses the bounds columns to indicate the constant parameters. The last table in the figure shows the final parameter estimates. The estimates of all free parameters are significantly different than 0. As expected, the final estimates of the constant parameters *Xr* and *Pn* have not changed from their initial values.

Output 5.3.2 Detailed Results for the LOGNGPD Distribution

```
                         The HPSEVERITY Procedure
                           logngpd Distribution

                         Distribution Information

Name                                                                logngpd
Description              Lognormal Body-GPD Tail Distribution. Mu, Sigma, and Xi
                         are free parameters. Xr and Pn are constant parameters.
Distribution                                                              5
Parameters


                   Initial Parameter Values and Bounds

                            Initial         Lower         Upper
              Parameter       Value         Bound         Bound

              Mu            1.49954        -Infty         Infty
              Sigma         0.76306    1.05367E-8         Infty
              Xi            0.36661    1.05367E-8         Infty
              Xr            1.27395      Constant      Constant
              Pn            0.80000      Constant      Constant


                            Convergence Status

                 Convergence criterion (GCONV=1E-8) satisfied.


                           Optimization Summary

                     Optimization Technique    Trust Region
                     Iterations                          11
                     Function Calls                      33
                     Failed Function Calls                1
                     Log Likelihood              -209.39116


                           Parameter Estimates

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

          Mu              1.57921       0.06426      24.57      <.0001
          Sigma           0.31868       0.04459       7.15      <.0001
          Xi              1.03771       0.38205       2.72      0.0078
          Xr              1.27395      Constant        .          .
          Pn              0.80000      Constant        .          .
```

The following SAS statements use the parameter estimates to compute the value where the tail region is estimated to start ($x_b = e^{\hat{\mu}}\hat{x_r}$) and the scale of the GPD tail distribution ($\theta_t = \frac{G(x_b)}{g(x_b)}\frac{(1-p_n)}{p_n}$):

```
/*-------- Compute tail cutoff and tail distribution's scale --------*/
data xb_thetat(keep=x_b theta_t);
   set parmest(where=(_MODEL_='logngpd' and _TYPE_='EST'));
   x_b = exp(Mu) * Xr;
   theta_t = (CDF('LOGN',x_b,Mu,Sigma)/PDF('LOGN',x_b,Mu,Sigma)) *
             ((1-Pn)/Pn);
run;

proc print data=xb_thetat noobs;
run;
```

Output 5.3.3 Start of the Tail and Scale of the GPD Tail Distribution

```
    x_b      theta_t

6.18005      1.27865
```

The computed values of x_b and θ_t are shown as *x_b* and *theta_t* in Output 5.3.3. Equipped with this additional derived information, you can now interpret the results of fitting the mixed-tail distribution as follows:

- The tail starts at $y \approx 6.18$. The primary benefit of using the scale-normalized cutoff (x_r) as the constant parameter instead of using the actual cutoff (x_b) is that the absolute cutoff is optimized by virtue of optimizing the scale of the body region ($\theta = e^{\mu}$).
- The values $y \leq 6.18$ follow the lognormal distribution with parameters $\mu \approx 1.58$ and $\sigma \approx 0.32$. These parameter estimates are reasonably close to the parameters used for simulating the sample.
- The values $y_t = y - 6.18$ ($y_t > 0$) follow the GPD distribution with scale $\theta_t \approx 1.28$ and shape $\xi \approx 1.04$.

Example 5.4: Fitting a Scaled Tweedie Model with Regressors

The Tweedie distribution is often used in the insurance industry to explain the effect of independent variables (regressors) on the distribution of losses. PROC HPSEVERITY provides a predefined scaled Tweedie distribution (STWEEDIE) that enables you to model the regression effects on the scale parameter. The scale regression model has its own advantages such as the ability to easily account for inflation effects. This example illustrates how that model can be used to evaluate the effect of regressors on the *mean* of the Tweedie distribution, which is useful in problems such rate-making and pure premium modeling.

Assume a Tweedie process, whose mean μ is affected by k regressors x_j, $j = 1, \ldots, k$ as follows:

$$\mu = \mu_0 \exp\left(\sum_{j=1}^{k} \beta_j x_j\right)$$

where μ_0 represents the base value of the mean (you can think of μ_0 as $\exp(\beta_0)$, where β_0 is the intercept). This model for the mean is identical to the popular generalized linear model for the mean with a logarithmic link function.

More interestingly, it parallels the model used by PROC HPSEVERITY for the scale parameter θ,

$$\theta = \theta_0 \exp\left(\sum_{j=1}^{k} \beta_j x_j\right)$$

where θ_0 represents the base value of the scale parameter. As described in the section "Tweedie Distributions" on page 161, for the parameter range $p \in (1, 2)$, the mean of the Tweedie distribution is given by

$$\mu = \theta\lambda\frac{2-p}{p-1}$$

where λ is the Poisson mean parameter of the scaled Tweedie distribution. This relationship enables you to use the scale regression model to infer the effect of regressors on the mean of the distribution.

Let the data set Work.Test_Sevtw contain a sample generated from a Tweedie distribution with dispersion parameter $\phi = 0.5$, index parameter $p = 1.75$, and the mean parameter that is affected by three regression variables x1, x2, and x3 as follows:

$$\mu = 5\ \exp(0.25\,\text{x1} - \text{x2} + 3\,\text{x3})$$

Thus, the population values of regression parameters are $\mu_0 = 5$, $\beta_1 = 0.25$, $\beta_2 = -1$, and $\beta_3 = 3$. You can find the code used to generate the sample in the PROC HPSEVERITY sample program *hpseve04.sas*.

The following PROC HPSEVERITY step uses the sample in Work.Test_Sevtw data set to estimate the parameters of the scale regression model for the predefined scaled Tweedie distribution (STWEEDIE) with the dual quasi-Newton (QUANEW) optimization technique:

```
proc hpseverity data=test_sevtw outest=estw covout print=all;
   loss y;
   scalemodel x1-x3;

   dist stweedie;
   nloptions tech=quanew;
run;
```

The dual quasi-Newton technique is used because it requires only the first-order derivatives of the objective function, and it is harder to compute reasonably accurate estimates of the second-order derivatives of Tweedie distribution's PDF with respect to the parameters.

Some of the key results prepared by PROC HPSEVERITY are shown in Output 5.4.1 and Output 5.4.2. The distribution information and the convergence results are shown in Output 5.4.1.

Output 5.4.1 Convergence Results for the STWEEDIE Model with Regressors

```
                         The HPSEVERITY Procedure
                          stweedie Distribution

                         Distribution Information

Name                                                         stweedie
Description                  Tweedie Distribution with Scale Parameter
Distribution Parameters                                             3
Regression Parameters                                               3


                            Convergence Status

       Convergence criterion (FCONV=2.220446E-16) satisfied.


                           Optimization Summary

            Optimization Technique    Dual Quasi-Newton
            Iterations                               42
            Function Calls                          218
            Log Likelihood                      -1044.3
```

The final parameter estimates of the STWEEDIE regression model are shown in Output 5.4.2. The estimate that is reported for the parameter Theta is the estimate of the base value θ_0. The estimates of regression coefficients β_1, β_2, and β_3 are indicated by the rows of x1, x2, and x3, respectively.

Output 5.4.2 Parameter Estimates for the STWEEDIE Model with Regressors

```
                           Parameter Estimates

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

    Theta            0.82888      0.26657        3.11      0.0021
    Lambda          16.57174     13.12083        1.26      0.2076
    P                1.75440      0.20187        8.69      <.0001
    x1               0.27970      0.09876        2.83      0.0049
    x2              -0.76715      0.10313       -7.44      <.0001
    x3               3.03225      0.10142       29.90      <.0001
```

If your goal is to explain the effect of regressors on the scale parameter, then the output displayed in Output 5.4.2 is sufficient. But, if you want to compute the effect of regressors on the mean of the distribution, then you need to do some postprocessing. Using the relationship between μ and θ, μ can be written in terms of the parameters of the STWEEDIE model as

$$\mu = \theta_0 \exp\left(\sum_{j=1}^{k} \beta_j x_j\right) \lambda \frac{2-p}{p-1}$$

This shows that the parameters β_j are identical for the mean and the scale model, and the base value μ_0 of the mean model is

$$\mu_0 = \theta_0 \lambda \frac{2-p}{p-1}$$

The estimate of μ_0 and the standard error associated with it can be computed by using the property of the functions of maximum likelihood estimators (MLE). If $g(\Omega)$ represents a totally differentiable function of parameters Ω, then the MLE of g has an asymptotic normal distribution with mean $g(\hat{\Omega})$ and covariance $C = (\partial \mathrm{g})'\Sigma(\partial \mathrm{g})$, where $\hat{\Omega}$ is the MLE of Ω, Σ is the estimate of covariance matrix of Ω, and $\partial \mathrm{g}$ is the gradient vector of g with respect to Ω evaluated at $\hat{\Omega}$. For μ_0, the function is $g(\Omega) = \theta_0\lambda(2-p)/(p-1)$. The gradient vector is

$$\begin{aligned} \partial \mathrm{g} &= \left(\frac{\partial g}{\partial \theta_0} \quad \frac{\partial g}{\partial \lambda} \quad \frac{\partial g}{\partial p} \quad \frac{\partial g}{\partial \beta_1} \cdots \frac{\partial g}{\partial \beta_k} \right) \\ &= \left(\frac{\mu_0}{\theta_0} \quad \frac{\mu_0}{\lambda} \quad \frac{-\mu_0}{(p-1)(2-p)} \quad 0 \ldots 0 \right) \end{aligned}$$

You can write a DATA step that implements these computations by using the parameter and covariance estimates prepared by PROC HPSEVERITY step. The DATA step program is available in the sample program *sevex04.sas*. The estimates of μ_0 prepared by that program are shown in Output 5.4.3. These estimates and the estimates of β_j as shown in Output 5.4.2 are reasonably close (that is, within one or two standard errors) to the parameters of the population from which the sample in Work.Test_Sevtw data set was drawn.

Output 5.4.3 Estimate of the Base Value Mu0 of the Mean Parameter

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

          Mu0        4.47179      0.42225     10.5904          0
```

Another effect of using the scaled Tweedie distribution to model the regression effects is that the regressors also affect the variance V of the Tweedie distribution. The variance is related to the mean as $V = \phi\mu^p$, where ϕ is the dispersion parameter. Using the relationship between the parameters TWEEDIE and STWEEDIE distributions as described in the section “Tweedie Distributions” on page 161, the regression model for the dispersion parameter is

$$\begin{aligned} \log(\phi) &= (2-p)\log(\mu) - \log(\lambda(2-p)) \\ &= ((2-p)\log(\mu_0) - \log(\lambda(2-p))) + (2-p)\sum_{j=1}^{k} \beta_j x_j \end{aligned}$$

Subsequently, the regression model for the variance is

$$\begin{aligned} \log(V) &= 2\log(\mu) - \log(\lambda(2-p)) \\ &= (2\log(\mu_0) - \log(\lambda(2-p))) + 2\sum_{j=1}^{k} \beta_j x_j \end{aligned}$$

In summary, PROC HPSEVERITY enables you to estimate regression effects on various parameters and statistics of the Tweedie model.

Example 5.5: Fitting Distributions to Interval-Censored Data

In some applications, the data available for modeling might not be exact. A commonly encountered scenario is the use of grouped data from an external agency, which for several reasons, including privacy, does not provide information about individual loss events. The losses are grouped into disjoint bins, and you know only the range and number of values in each bin. Each group is essentially interval-censored, because you know that a loss magnitude is in certain interval, but you do not know the exact magnitude. This example illustrates how you can use PROC HPSEVERITY to model such data.

The following DATA step generates sample grouped data for dental insurance claims, which is taken from Klugman, Panjer, and Willmot (1998):

```
/* Grouped dental insurance claims data
     (Klugman, Panjer, and Willmot, 1998) */
data gdental;
   input lowerbd upperbd count @@;
   datalines;
0 25 30   25 50 31   50 100 57   100 150 42   150 250 65   250 500 84
500 1000 45   1000 1500 10   1500 2500 11   2500 4000 3
;
run;
```

The following PROC HPSEVERITY step fits all the predefined distributions to the data in Work.Gdental data set:

```
/* Fit all predefined distributions */
proc hpseverity data=gdental edf=turnbull print=all criterion=aicc;
   loss / rc=lowerbd lc=upperbd;
   weight count;
   dist _predef_;
   performance nthreads=1;
run;
```

The EDF= option in the PROC HPSEVERITY statement specifies that the Turnbull's method be used for EDF estimation. The LOSS statement specifies the left and right boundaries of each group as the right-censoring and left-censoring limits, respectively. The variable count records the number of losses in each group and is specified in the WEIGHT statement. Note that no response variable is specified in the LOSS statement, which is allowed as long as each observation in the input data set is censored. The PERFORMANCE statement specifies that just one thread of execution be used, to minimize the overhead associated with multithreading, because the input data set is very small.

Some of the key results prepared by PROC HPSEVERITY are shown in Output 5.5.1. According to the "Model Selection" table in Output 5.5.1, all distribution models have converged. The "All Fit Statistics" table in Output 5.5.1 indicates that the exponential distribution (EXP) has the best fit for data according to a majority of the likelihood-based statistics.

Output 5.5.1 Statistics of Fit for Interval-Censored Data

```
                        The HPSEVERITY Procedure

                             Input Data Set

                           Name    WORK.GDENTAL


                             Model Selection

              Distribution    Converged         AICC    Selected

              Burr            Yes           51.41112    No
              Exp             Yes           44.64768    Yes
              Gamma           Yes           47.63969    No
              Igauss          Yes           48.05874    No
              Logn            Yes           47.34027    No
              Pareto          Yes           47.16908    No
              Gpd             Yes           47.16908    No
              Weibull         Yes           47.47700    No


                           All Fit Statistics

                    -2 Log
Distribution    Likelihood          AIC         AICC          BIC           KS

Burr             41.41112*     47.41112     51.41112     48.31888     0.08974*
Exp              42.14768      44.14768*    44.64768*    44.45026*    0.26412
Gamma            41.92541      45.92541     47.63969     46.53058     0.19569
Igauss           42.34445      46.34445     48.05874     46.94962     0.34514
Logn             41.62598      45.62598     47.34027     46.23115     0.16853
Pareto           41.45480      45.45480     47.16908     46.05997     0.11423
Gpd              41.45480      45.45480     47.16908     46.05997     0.11423
Weibull          41.76272      45.76272     47.47700     46.36789     0.17238

                Note: The asterisk (*) marks the best model
                   according to each column's criterion.

                            All Fit Statistics

                   Distribution           AD           CvM

                   Burr              0.00103*    0.0000816*
                   Exp               0.09936       0.01866
                   Gamma             0.04608       0.00759
                   Igauss            0.12301       0.02562
                   Logn              0.01884       0.00333
                   Pareto            0.00739     0.0009084
                   Gpd               0.00739     0.0009084
                   Weibull           0.03293       0.00472

                    Note: The asterisk (*) marks the best
                           model according to each
                             column's criterion.
```

Example 5.6: Benefits of Distributed and Multithreaded Computing

One of the key features of the HPSEVERITY procedure is that is takes advantage of the distributed and multithreaded computing machinery in order to solve a given problem faster. This example illustrates the benefits of using multithreading and distributed computing.

The example uses a simulated data set Work.Largedata, which contains 10,000,000 observations, some of which are right-censored or left-truncated. The losses are affected by three external effects. The DATA step program that generates this data set is available in the accompanying sample program *hpseve06.sas*.

The following PROC HPSEVERITY step fits all the predefined distributions to the data in Work.Largedata data set on the client machine with just one thread of computation:

```
/* Fit all predefined distributions without any multithreading or
   distributed computing */
proc hpseverity data=largedata criterion=aicc initsample(size=20000);
   loss y / lt=threshold rc=limit;
   scalemodel x1-x3;
   dist _predef_;
   performance nthreads=1 bufsize=1000000 details;
run;
```

The NTHREADS=1 option in the PERFORMANCE statement specifies that just one thread of computation be used. The absence of the NODES= option in the PERFORMANCE statement specifies that single-machine mode of execution be used. That is, this step does not use any multithreading or distributed computing. The BUFSIZE= option in the PERFORMANCE statement specifies the number of observations to read at one time. Specifying a larger value tends to decrease the time it takes to load the data. The DETAILS option in the performance statement enables reporting of the timing information. The INITSAMPLE option in the PROC HPSEVERITY statement specifies that a uniform random sample of maximum 20,000 observations be used for parameter initialization.

The “Performance Information” and “Procedure Task Timing” tables that PROC HPSEVERITY prepares are shown in Output 5.6.1. The “Performance Information” table contains the information about the execution environment. The “Procedure Task Timing” table indicates the total time and relative time taken by each of the four main steps of PROC HPSEVERITY. As that table shows, it takes around 25 minutes for the task of estimating parameters, which is usually the most time-consuming of all the tasks.

Output 5.6.1 Performance for Single-Machine Mode with No Multithreading

```
                    The HPSEVERITY Procedure

                    Performance Information

              Execution Mode        Single-Machine
              Number of Threads     1
```

Output 5.6.1 *continued*

```
                          Procedure Task Timing

      Task                                      Seconds     Percent

      Load and Prepare Models                      4.41       0.29%
      Load and Prepare Data                        1.36       0.09%
      Initialize Parameters                        0.81       0.05%
      Estimate Parameters                       1513.85      99.48%
      Compute Fit Statistics                       1.26       0.08%
```

If the grid appliance is not available, you can improve the performance by using multiple threads of computation; this is in fact the default. The following PROC HPSEVERITY step fits all the predefined distributions by using all the logical CPU cores of the machine:

```
options cpucount=actual;

/* Fit all predefined distributions with multithreading, but no
   distributed computing */
proc hpseverity data=largedata criterion=aicc initsample(size=20000);
   loss y / lt=threshold rc=limit;
   scalemodel x1-x3;
   dist _predef_;
   performance bufsize=1000000 details;
run;
```

When you do not specify the NTHREADS= option in the PERFORMANCE statement, the HPSEVERITY procedure uses the value of the CPUCOUNT= system option to decide the number of threads to use in single-machine mode. Setting the CPUCOUNT= option to ACTUAL before the PROC HPSEVERITY step enables the procedure to use all the logical cores of the machine. The machine that is used to obtain these results (and the earlier results in Output 5.6.1) has four physical CPU cores, each with a clock speed of 3.4 GHz. Hyperthreading is enabled on the CPUs to yield eight logical CPU cores; this number is confirmed by the “Performance Information” table in Output 5.6.2. The results in the “Procedure Task Timing” table in Output 5.6.2 indicate that the use of multithreading has improved the performance significantly by reducing the time to estimate parameters to around 5.5 minutes.

Output 5.6.2 Performance for Single-Machine Mode with Eight Threads

```
                         The HPSEVERITY Procedure

                         Performance Information

                    Execution Mode        Single-Machine
                    Number of Threads     8
```

Output 5.6.2 *continued*

```
                        Procedure Task Timing

        Task                                  Seconds     Percent

        Load and Prepare Models                  0.34       0.10%
        Load and Prepare Data                    1.01       0.30%
        Initialize Parameters                    0.65       0.19%
        Estimate Parameters                    335.37      99.14%
        Compute Fit Statistics                   0.89       0.26%
```

When a grid appliance is available, performance can be further improved by using more than one node in the distributed mode of execution. Large data sets are usually predistributed on the grid appliance that hosts a distributed database. In other words, large problems are best suited for the alongside-the-database model of execution. However, for the purpose of illustration, this example assumes that the data set is available on the client machine and is then distributed to the grid nodes by the HPSEVERITY procedure according to the options that are specified in the PERFORMANCE statement.

The next few PROC HPSEVERITY steps are run on a grid appliance by varying the number of nodes and the number of threads that are used within each node.

First, the following statements are submitted to specify the appliance host (GRIDHOST= system option) and the installation location of shared libraries on the appliance (GRIDINSTALLLOC= system option):

```
option set=GRIDHOST      ="&GRIDHOST";
option set=GRIDINSTALLLOC="&GRIDINSTALLLOC";
```

To run the preceding statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to macro variables with the appropriate values. For more information about the GRIDHOST= and GRIDINSTALLLOC= options, see the section "PERFORMANCE Statement" on page 34.

To establish a reference point for the performance of one CPU of a grid node, the results of using only one node of the grid appliance without any multithreading are presented first. The particular grid appliance that is used to obtain these results has eight nodes. Each node has 24 logical CPU cores with a clock speed of 2.93 GHz. The following PROC HPSEVERITY step fits all the predefined distributions to the data in the Work.Largedata data set:

```
/* Fit all predefined distributions on 1 grid node without
   any multithreading */
proc hpseverity data=largedata criterion=aicc initsample(size=20000);
   loss y / lt=threshold rc=limit;
   scalemodel x1-x3;
   dist _predef_;
   performance nthreads=1 nodes=1 details;
run;
```

The PERFORMANCE statement specifies that only one node be used to fit the models, with only one thread of computation on that node. The "Performance Information" and "Procedure Task Timing" tables that PROC HPSEVERITY prepares are shown in Output 5.6.3. It takes around 36 minutes to complete the task of estimating parameters.

The computations and time taken to fit each model are also shown in the “Estimation Details” table, which is generated whenever you specify the DETAILS option in the PERFORMANCE statement. This table can be useful for comparing the relative effort required to fit each model and drawing some broader conclusions. For example, even if the Pareto distribution takes a larger number of iterations, function calls, and gradient and Hessian updates than the gamma distribution, it takes less time to complete; this indicates that the individual PDF and CDF computations of the gamma distribution are more expensive than those of the Pareto distribution.

Output 5.6.3 Performance on One Grid Appliance Node with No Multithreading

```
                          The HPSEVERITY Procedure

                          Performance Information

                 Host Node                     << your grid host >>
                 Execution Mode                Distributed
                 Grid Mode                     Symmetric
                 Number of Compute Nodes       1
                 Number of Threads per Node    1


                             Estimation Details
                                        Function  Gradient   Hessian       Time
   Distribution  Converged  Iterations     Calls   Updates   Updates  (Seconds)

   Burr          Yes                11        28       104        90     290.37
   Exp           Yes                 4        12        27        20      29.98
   Gamma         Yes                 5        15        35        27     777.45
   Igauss        Yes                 4        12        27        20     271.45
   Logn          Yes                 4        12        27        20     114.95
   Pareto        Maybe              50       137      1430      1377     461.36
   Gpd           Yes                 6        17        44        35     116.90
   Weibull       Yes                 4        12        27        20      70.84


                           Procedure Task Timing

         Task                                  Seconds     Percent

         Load and Prepare Models                  0.48       0.02%
         Load and Prepare Data                    0.70       0.03%
         Initialize Parameters                    1.22       0.06%
         Estimate Parameters                   2133.31      99.80%
         Compute Fit Statistics                   1.91       0.09%
```

To obtain the next reference point for performance, the following PROC HPSEVERITY step specifies that 24 computation threads be used on one node of the grid appliance:

```
/* Fit all predefined distributions on 1 grid node with multithreading */
proc hpseverity data=largedata criterion=aicc initsample(size=20000);
   loss y / lt=threshold rc=limit;
   scalemodel x1-x3;
   dist _predef_;
   performance nthreads=24 nodes=1 details;
run;
```

The performance tables that are prepared by the preceding statements are shown in Output 5.6.4. As the "Procedure Task Timing" table shows, use of multithreading has improved the performance significantly over that of the single-threaded case. Now, it takes around 3 minutes to complete the task of estimating parameters.

Output 5.6.4 Performance Information with Multithreading but No Distributed Computing

```
                          The HPSEVERITY Procedure

                          Performance Information

               Host Node                       << your grid host >>
               Execution Mode                  Distributed
               Grid Mode                       Symmetric
               Number of Compute Nodes         1
               Number of Threads per Node      24


                           Procedure Task Timing

       Task                                    Seconds     Percent

       Load and Prepare Models                    0.36       0.20%
       Load and Prepare Data                      0.39       0.21%
       Initialize Parameters                      0.98       0.53%
       Estimate Parameters                      181.10      98.40%
       Compute Fit Statistics                     1.21       0.66%
```

You can combine the power of multithreading and distributed computing by specifying that multiple nodes of the grid and all available threads of execution within each node be used to accomplish the task. The following PROC HPSEVERITY step specifies that all eight nodes of the grid appliance be used:

```
/* Fit all predefined distributions with distributed computing and
   multithreading within each node */
proc hpseverity data=largedata criterion=aicc initsample(size=20000);
   loss y / lt=threshold rc=limit;
   scalemodel x1-x3;
   dist _predef_;
   performance nodes=8 details;
run;
```

Omitting the NTHREADS= option from the PERFORMANCE statement in distributed mode results in the use of all 24 logical CPU cores on each node of the grid.

When the DATA= data set is local to the client machine, as it is in this example, you must specify a nonzero value for the NODES= option in the PERFORMANCE statement in order to enable the distributed mode of execution. In other words, for the distributed mode that is not executing alongside the database, omitting the NODES= option is equivalent to specifying NODES=0, which is single-machine mode.

The performance tables that are prepared by the preceding statements are shown in Output 5.6.5. If you compare these tables to the tables in Output 5.6.3 and Output 5.6.4, you see that the task that would have taken a long time with a single thread of execution on a single machine can be performed in a much shorter time by using the full computational resources of the grid appliance to combine the power of multithreaded and distributed computing.

Output 5.6.5 Performance Information with Distributed Computing and Multithreading

```
                    The HPSEVERITY Procedure

                    Performance Information

           Host Node                     << your grid host >>
           Execution Mode                Distributed
           Grid Mode                     Symmetric
           Number of Compute Nodes       8
           Number of Threads per Node    24


                     Procedure Task Timing

   Task                                  Seconds     Percent

   Load and Prepare Models                  0.92       1.66%
   Load and Prepare Data                    0.08       0.14%
   Initialize Parameters                    1.23       2.22%
   Estimate Parameters                     51.80      93.25%
   Compute Fit Statistics                   1.52       2.73%
```

The machines that were used to obtain these performance results are relatively modest machines, and PROC HPSEVERITY was executed in a multiuser environment; that is, background processes were running in single-machine mode or other users were using the grid in distributed mode. For time-critical applications, you can use a larger, dedicated grid that consists of more powerful machines to achieve more dramatic performance improvement.

Example 5.7: Estimating Parameters Using Cramér-von Mises Estimator

PROC HPSEVERITY enables you to estimate model parameters by minimizing your own objective function. This example illustrates how you can use PROC HPSEVERITY to implement the Cramér-von Mises estimator. Let $F(y_i;\Theta)$ denote the estimate of CDF at y_i for a distribution with parameters Θ, and let $F_n(y_i)$ denote the empirical estimate of CDF (EDF) at y_i that is computed from a sample y_i, $1 \leq i \leq N$. Then, the Cramér-von Mises estimator of the parameters is defined as

$$\hat{\Theta} = \arg\min_{\Theta} \sum_{i=1}^{N} (F(y_i;\Theta) - F_n(y_i))^2$$

This estimator belongs to the class of minimum distance estimators. It attempts to estimate the parameters such that the squared distance between the CDF and EDF estimates is minimized.

The following PROC HPSEVERITY step uses the Cramér-von Mises estimator to fit four candidate distribution models, including the LOGNGPD mixed-tail distribution model that was defined in "Example 5.3: Defining a Model for Mixed-Tail Distributions" on page 227. The input sample is the same as is used in that example.

```
options cmplib=(work.sevexmpl);

proc hpseverity data=testmixdist obj=cvmobj print=all;
   loss y;
   dist logngpd burr logn gpd;

   * Cramer-von Mises estimator (minimizes the distance *
   * between parametric and nonparametric estimates)    *;
   cvmobj = _cdf_(y);
   cvmobj = (cvmobj -_edf_(y))**2;
run;
```

The OBJ= option in the PROC HPSEVERITY statement specifies that the objective function cvmobj should be minimized. The programming statements compute the contribution of each observation in the input data set to the objective function cvmobj. The use of keyword functions _CDF_ and _EDF_ makes the program applicable to all the distributions.

Some of the key results prepared by PROC HPSEVERITY are shown in Output 5.7.1. The "Model Selection" table indicates that all models converged. When you specify a custom objective function, the default selection criterion is the value of the custom objective function. The "All Fit Statistics" table indicates that LOGNGPD is the best distribution according to all the statistics of fit. Comparing the fit statistics of Output 5.7.1 with those of Output 5.3.1 indicates that the use of the Cramér-von Mises estimator has resulted in smaller values for all the EDF-based statistics of fit for all the models, which is expected from a minimum distance estimator.

Output 5.7.1 Summary of Cramér-von Mises Estimation

```
                    The HPSEVERITY Procedure

                         Input Data Set

             Name                      WORK.TESTMIXDIST
             Label    Lognormal Body-GPD Tail Sample


                         Model Selection

       Distribution    Converged        cvmobj    Selected

       logngpd         Yes             0.02694    Yes
       Burr            Yes             0.03325    No
       Logn            Yes             0.03633    No
       Gpd             Yes             2.96090    No
```

Output 5.7.1 *continued*

```
                               All Fit Statistics

                                  -2 Log
Distribution        cvmobj    Likelihood           AIC          AICC           BIC

logngpd           0.02694*    419.49635*    429.49635*    430.13464*    442.52220*
Burr              0.03325     436.58823     442.58823     442.83823     450.40374
Logn              0.03633     491.88659     495.88659     496.01030     501.09693
Gpd               2.96090     560.35409     564.35409     564.47780     569.56443

                 Note: The asterisk (*) marks the best model
                    according to each column's criterion.

                               All Fit Statistics

           Distribution            KS              AD             CvM

           logngpd           0.51332*        0.21563*        0.03030*
           Burr              0.53084         0.82875         0.03807
           Logn              0.52469         2.08312         0.04173
           Gpd               2.99095        15.51378         2.97806

                 Note: The asterisk (*) marks the best model
                    according to each column's criterion.
```

References

D'Agostino, R. B. and Stephens, M., eds. (1986), *Goodness-of-Fit Techniques*, New York: Marcel Dekker.

Danielsson, J., De Haan, L., Peng, L., and de Vries, C. G. (2001), "Using a Bootstrap Method to Choose the Sample Fraction in Tail Index Estimation," *Journal of Multivariate Analysis*, 76, 226–248.

Dunn, P. K. and Smyth, G. K. (2005), "Series Evaluation of Tweedie Exponential Dispersion Model Densities," *Statistics and Computing*, 15, 267–280.

Frydman, H. (1994), "A Note on Nonparametric Estimation of the Distribution Function from Interval-Censored and Truncated Observations," *Journal of the Royal Statistical Society, Series B*, 56, 71–74.

Gentleman, R. and Geyer, C. J. (1994), "Maximum Likelihood for Interval Censored Data: Consistency and Computation," *Biometrika*, 81, 618–623.

Hill, B. M. (1975), "A Simple General Approach to Inference about the Tail of a Distribution," *Annals of Statistics*, 3, 1163–1173.

Jørgensen, B. (1987), "Exponential Dispersion Models (with discussion)," *Journal of the Royal Statistical Society, Series B*, 49, 127–162.

Kaplan, E. L. and Meier, P. (1958), "Nonparametric Estimation from Incomplete Observations," *Journal of the American Statistical Association*, 53, 457–481.

Klein, J. P. and Moeschberger, M. L. (1997), *Survival Analysis: Techniques for Censored and Truncated Data*, New York: Springer-Verlag.

Klugman, S. A., Panjer, H. H., and Willmot, G. E. (1998), *Loss Models: From Data to Decisions*, New York: John Wiley & Sons.

Koziol, J. A. and Green, S. B. (1976), "A Cramér–von Mises Statistic for Randomly Censored Data," *Biometrika*, 63, 466–474.

Lai, T. L. and Ying, Z. (1991), "Estimating a Distribution Function with Truncated and Censored Data," *Annals of Statistics*, 19, 417–442.

Lynden-Bell, D. (1971), "A Method of Allowing for Known Observational Selection in Small Samples Applied to 3CR Quasars," *Monthly Notices of the Royal Astronomical Society*, 155, 95–118.

Turnbull, B. W. (1976), "The Empirical Distribution Function with Arbitrarily Grouped, Censored, and Truncated Data," *Journal of the Royal Statistical Society, Series B*, 38, 290–295.

Tweedie, M. C. K. (1984), "An Index Which Distinguishes between Some Important Exponential Families," in J. K. Ghosh and J. Roy, eds., *Statistics: Applications and New Directions—Proceedings of the Indian Statistical Institute Golden Jubilee International Conference*, 579–604.

Subject Index

Syntax Index

Your Turn