Language Concepts

This section highlights several concepts and tools that are useful with base SAS procedures.

Temporary and Permanent SAS Data Sets

SAS data sets can have a one-level name or a two-level name. Typically, names of temporary SAS data sets have only one level and are stored in the WORK data library. The WORK data library is defined automatically at the beginning of the SAS session and is automatically deleted at the end of the SAS session. Procedures assume that SAS data sets that are specified with a one-level name are to be read from or written to the WORK data library, unless you specify a USER data library (see “USER Data Library” on page 16). For example, the following PROC PRINT steps are equivalent. The second PROC PRINT step assumes that the DEBATE data set is in the WORK data library:

```
proc print data=work.debate;
run;
```
The SAS system options WORK=, WORKINIT, and WORKTERM affect how you work with temporary and permanent libraries. See SAS Language Reference: Dictionary for complete documentation.

Typically, two-level names represent permanent SAS data sets. A two-level name takes the form libref.SAS-data-set. The libref identifies an external storage location that stores SAS data sets in your operating environment. A LIBNAME statement associates a libref with an external storage location. In the following PROC PRINT step, PROCLIB is the libref and EMP is the SAS data set within the library:

```
libname proclib 'SAS-data-library';
proc print data=proclib.emp;
run;
```

**USER Data Library**

You can use one-level names for permanent SAS data sets by specifying a USER data library. You can assign a USER data library with a LIBNAME statement or with the SAS system option USER=. After you specify a USER data library, the procedure assumes that data sets with one-level names are in the USER data library instead of the WORK data library. For example, the following PROC PRINT step assumes that DEBATE is in the USER data library:

```
options user='SAS-data-library';
proc print data=debate;
run;
```

Note: If you have a USER data library defined, you can still use the WORK data library by specifying WORK.SAS-data-set.

---

**SAS System Options**

Some SAS system option settings affect procedure output. The following are the SAS system options that you are most likely to use with SAS procedures:

- **BYLINE|NOBYLINE**
- **DATE|NODATE**
- **DETAILS|NODETAILS**
- **FMTERR|NOFMTERR**
- **FORMCHAR=**
- **FORMDLIM=**
- **LABEL|NOLABEL**
- **LINESIZE=**
- **NUMBER|NONUMBER**
- **PAGENO=**
- **PAGESIZE=**
- **REPLACE|NOREPLACE**
- **SOURCE|NOSOURCE**

For a complete description of SAS system options, see SAS Language Reference: Dictionary.
Data Set Options

Most of the procedures that read data sets or create output data sets accept data set options. SAS data set options appear in parentheses after the data set specification. Here is an example:

```
proc print data=stocks(obs=25 pw=green);
```

The individual procedure chapters contain reminders that you can use data set options where it is appropriate.

SAS data set options are

- ALTER=
- LABEL=
- BUFNO=
- OBS=
- BUFSIZE=
- OUTREP=
-_CNTLLEV=
- PW=
- COMPRESS=
- PWREQ=
- DLMGACTION=
- READ=
- DROP=
- RENAME=
- ENCRYPT=
- REPLACE=
- FILECLOSE=
- REUSE=
- FILEFMT=
- SORTEDBY=
- FIRSTOBS=
- TRANTAB=
- GENMAX=
- TYPE=
- GENNUM=
- WHERE=
- IN=
- WHEREUP=
- INDEX=
- WRITE=
- KEEP=

For a complete description of SAS data set options, see SAS Language Reference: Dictionary.
Global Statements

You can use these global statements anywhere in SAS programs except after a DATALINES, CARDS, or PARMCARDS statement:

- comment
- ODS
- DM
- OPTIONS
- ENDSAS
- PAGE
- FILENAME
- RUN
- FOOTNOTE
- %RUN
- %INCLUDE
- SKIP
- LIBNAME
- TITLE
- %LIST
- X
- MISSING

For information on all but the ODS statement, refer to SAS Language Reference: Dictionary. For some information on the ODS statement, refer to “Output Delivery System” on page 18 and to The Complete Guide to the SAS Output Delivery System.

Procedure Concepts

This section contains background information on concepts and tools that are common to many base SAS procedures.

Input Data Sets

Many base procedures require an input SAS data set. You specify the input SAS data set using the DATA= option in the procedure statement, for example,

```
proc print data=emp;
```

If you omit the DATA= option, the procedure uses the value of the SAS system option _LAST_. The default of _LAST_= is the most recently created SAS data set in the current SAS job or session. _LAST_= is described in detail in SAS Language Reference: Dictionary.

Output Delivery System

Prior to Version 7, SAS procedures that produced printed output (that is, output that was destined for the procedure output file) generated output that was designed for a traditional line-printer. This type of output has limitations that prevent users from getting the most value from their results:

- Traditional SAS output is limited to monospace fonts. In this day of desktop document editors and publishing systems, users want more versatility in printed output.
Traditional SAS output provides no way for you to parse its contents. You cannot, for example, know in advance in what column the values for the third variable in a report begin.

Quite a few commonly used procedures did not produce output data sets. Users who wanted to use output from one of these procedures as input to another procedure relied on PROC PRINTTO and the DATA step to retrieve results that could not be stored in an output data set.

Beginning with Version 7, procedure output is much more flexible. The Output Delivery System (ODS) has been designed to overcome the limitations of traditional SAS output and to make it easy to make new formatting options available to users. ODS is a method of delivering output in a variety of formats and of making the formatted output easy to access. Important features of ODS include the following:

- ODS combines raw data with one or more table definitions to produce one or more output objects. These objects can be sent to any or all ODS destinations. The currently available ODS destinations can produce an output data set, traditional monospace output, and output that is formatted in Hyper Text Markup Language (HTML).
- ODS provides table definitions that define the structure of the output from procedures and from the DATA step. You can customize the output by modifying these definitions or by creating your own.
- ODS provides a way for you to choose individual output objects to send to ODS destinations. For instance, PROC UNIVARIATE produces five output objects. You can easily create HTML output, an output data set, or traditional Listing output from any or all of these output objects. You can send different output objects to different destinations.
- ODS stores a link to each output object in the Results folder in the Results window.

In addition, ODS removes responsibility for formatting output from individual procedures and from the DATA step. The procedure or DATA step supplies raw data and the name of the table definition that contains the formatting instructions, and ODS formats the output. Because formatting is now centralized in ODS, the addition of a new ODS destination does not affect any procedures or the DATA step. As future destinations are added to ODS, they will automatically become available to all procedures that support ODS and to the DATA step.

This section briefly illustrates these features. For more information about the Output Delivery System, see The Complete Guide to the SAS Output Delivery System.

**Storing Links in the Results Folder**

When you run a procedure that supports ODS, SAS automatically stores a link to the ODS output in the Results folder in the Results window.

Consider the following SAS program, which generates Listing output. The data set STATEPOP contains information about the distribution of the United States' population in metropolitan and nonmetropolitan areas for 1980 and 1990. A DATA step on page 1361 creates this data set.

```sas
options nodate pageno=1 linesize=80 pagesize=34;

proc univariate data=statepop mu0=3.5;
  var citypop_90 noncitypop_90;
  title;
```


Some of the Listing output appears in Output 2.1 on page 20. The Results folder (see Display 2.1 on page 22) shows the folders and output objects that the procedure produces.

**Output 2.1**  Partial Listing Output Produced by PROC UNIVARIATE

```plaintext
   The UNIVARIATE Procedure
   Variable: CityPop_90  (1990 metropolitan pop in millions)

   Moments
   N       51  Sum Weights   51
   Mean     3.87701961  Sum Observations  197.728
   Std Deviation  5.16465302  Variance  26.6736408
   Skewness     2.87109259  Kurtosis  10.537867
   Uncorrected SS 2100.27737  Corrected SS 1333.68204
   Coeff Variation 133.21194  Std Error Mean 0.72319608

   Basic Statistical Measures
   Location          Variability
   Mean     3.877020   Std Deviation      5.16465
   Median   2.423000   Variance        26.67364
   Mode     .          Range          28.66500
   Interquartile Range 3.60000

   Tests for Location: Mu0=3.5
   Test         -Statistic-      -----p Value-----
   Student’s t   t   0.521324      Pr > |t|   0.6044
   Sign         M     -9.5        Pr >= |M|  0.0110
   Signed Rank   S     -147       Pr >= |S|  0.1706
```
The UNIVARIATE Procedure
Variable: CityPop_90 (1990 metropolitan pop in millions)

Quantiles (Definition 5)

<table>
<thead>
<tr>
<th>Quantile</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>100% Max</td>
<td>28.799</td>
</tr>
<tr>
<td>99%</td>
<td>28.799</td>
</tr>
<tr>
<td>95%</td>
<td>14.166</td>
</tr>
<tr>
<td>90%</td>
<td>9.574</td>
</tr>
<tr>
<td>75% Q3</td>
<td>4.376</td>
</tr>
<tr>
<td>50% Median</td>
<td>2.423</td>
</tr>
<tr>
<td>25% Q1</td>
<td>0.776</td>
</tr>
<tr>
<td>10%</td>
<td>0.257</td>
</tr>
<tr>
<td>5%</td>
<td>0.191</td>
</tr>
<tr>
<td>1%</td>
<td>0.134</td>
</tr>
<tr>
<td>0% Min</td>
<td>0.134</td>
</tr>
</tbody>
</table>

Extreme Observations

<table>
<thead>
<tr>
<th>-----Lowest-----</th>
<th>-----Highest-----</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Obs</td>
</tr>
<tr>
<td>0.134</td>
<td>41</td>
</tr>
<tr>
<td>0.152</td>
<td>3</td>
</tr>
<tr>
<td>0.191</td>
<td>39</td>
</tr>
<tr>
<td>0.221</td>
<td>36</td>
</tr>
<tr>
<td>0.226</td>
<td>50</td>
</tr>
</tbody>
</table>
PROC UNIVARIATE generates a folder called Univariate in the Results folder. Within this folder are two more folders: one for each variable in the VAR statement. These folders contain links to the ODS output. Notice that the folder CityPop_90 contains five links while the folder NonCityPop_90 contains six. (It has additional output that reports on missing values.)

Creating HTML Output

In addition to creating Listing output, the Output Delivery System can create HTML output by formatting output objects in Hyper text Markup Language (HTML). You can browse these files with Internet Explorer, Netscape, or any other browser that fully supports the HTML 3.2 tag set.

The ODS HTML statement, which generates the HTML files, can create

- an HTML file (called the body file) that contains the results from the procedure
- a table of contents that links to the body file
- a table of pages that links to the body file
- a frame that displays the table of contents, the table of pages, and the body file.

For example, the ODS statement in the following SAS program generates four HTML files. ODS routes the results of the PROC UNIVARIATE step to the body file as well as to the Listing destination. ODS also creates the associated contents, page, and frame files.

Note: This example is for the UNIX operating environment. To successfully run the example in another operating environment, you may need to change the file specifications. See Appendix 4, “Alternate ODS HTML Statements for Running Examples in Different Operating Environments,” on page 1477.
/* Create HTML files. */
ods html file='body.htm'
    contents='contents.htm'
    page='page.htm'
    frame='frame.htm';

proc univariate data=statepop mu0=3.5;
    var citypop_90 noncitypop_90;
    title;
run;

/* Close the HTML destination. */
/* You must close this destination before */
/* you can browse the HTML files. */
ods html close;

The frame file appears in Display 2.2 on page 23.

Display 2.2  First View of the Frame File

<table>
<thead>
<tr>
<th>The UNIVARIATE Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable: CityPop_90 (1990 metropolitan pop in millions)</td>
</tr>
<tr>
<td>Moments</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>Mean</td>
</tr>
<tr>
<td>Std Deviation</td>
</tr>
<tr>
<td>Skewness</td>
</tr>
<tr>
<td>Uncorrected SS</td>
</tr>
<tr>
<td>Coeff Variation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Basic Statistical Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
</tr>
<tr>
<td>Mean</td>
</tr>
<tr>
<td>Median</td>
</tr>
<tr>
<td>Mode</td>
</tr>
</tbody>
</table>

If you click on Extreme Observations under CityPop_90 in the Table of Contents, the HTML table that contains that part of the procedure results appears at the top of the frame that contains the body file. (See Display 2.3 on page 24.)
For more information about creating HTML output, see the discussion of the ODS HTML Statement in Chapter 3, “The ODS Statements” in The Complete Guide to the SAS Output Delivery System. You can see many examples of HTML output in SAS Procedures Guide online documentation.

**Note**    Procedure options that affect presentation may not affect HTML output. For instance, the DOUBLE option in PROC PRINT, which inserts a blank line between observations, has no effect on HTML output.

### Identifying Output Objects

Several of the features of the Output Delivery System (such as selecting and excluding output objects to send to ODS destinations or creating a data set from an output object) require that you specify one or more particular output objects for the system to process. You identify an output object by its name, its label, or its path. To learn what these are, run your SAS program preceded by this statement:

```
ods trace on;
```

As long as the tracing feature is on, information about each output object that is created appears in the SAS log.

Use this statement to stop sending the information to the log:

```
ods trace off;
```

For example, the following SAS program produces the SAS log that is shown in Output 2.2 on page 25:

```
options nodate pageno=1 linesize=64 pagesize=60;
ods trace on;
proc univariate data=statepop mu0=3.5;
  var citypop_90 noncitypop_90;
title;
```
run;
ods trace off;

Output 2.2 SAS Log Produced by the ODS TRACE Statement

Compare the second output object that is created for CityPop_90 to the second output object that is created for NonCityPop_90. These objects are marked with an arrow (→). The names and labels of these objects are identical. Thus, using a name or a label can refer to multiple output objects, which is sometimes useful. If you want to reference each output object separately, you must use its path, which is unique.

```sas
options nodate pageno=1 linesize=64 pagesize=60;
ods trace on;
proc univariate data=statepop mu0=3.5;
var citypop_90 noncitypop_90;
title;
run;
```

Output Added:
--------------
Name: Moments
Label: Moments
Template: base.univariate.Moments
Path: Univariate.CityPop_90.Moments
--------------
Output Added:
--------------
Name: BasicMeasures  —
Label: Basic Measures of Location and Variability
Template: base.univariate.Measures
Path: Univariate.CityPop_90.BasicMeasures
--------------
Output Added:
--------------
Name: TestsForLocation
Label: Tests For Location
Template: base.univariate.Location
Path: Univariate.CityPop_90.TestsForLocation
--------------
Output Added:
--------------
Name: Quantiles
Label: Quantiles
Template: base.univariate.Quantiles
Path: Univariate.CityPop_90.Quantiles
--------------
Output Added:
--------------
Name: ExtremeObs
Label: Extreme Observations
Template: base.univariate.ExtObs
Path: Univariate.CityPop_90.ExtreameObs
--------------
If you compare this SAS log to the Results Folder that appears in Display 2.1 on page 22, you can see that the string that identifies the output in the Results folder is its label.

For more information about the trace record, see the discussion of the contents of the trace record in the documentation for the ODS TRACE statement in Chapter 3, “The ODS Statements” in *The Complete Guide to the SAS Output Delivery System*.

### Selecting Output Objects to Send to ODS Destinations

Some procedures, such as PROC UNIVARIATE, produce multiple output objects. Any procedure that uses ODS produces multiple output objects when you use BY-group processing. ODS enables you to select which of these output objects go to the open ODS destinations. (ODS destinations include the Listing destination, the HTML destination, and the Output destination. For more information about ODS destinations, see Chapter 1, “Basic Concepts about the Output Delivery System” in *The Complete Guide to the SAS Output Delivery System*. )
You choose the objects to send to destinations with the ODS SELECT or the ODS EXCLUDE statement. To select individual output objects, use this form of the ODS SELECT statement:

```
ODS SELECT selection(s);
```

where each value of selection can be a full path, a name, or a label (see the trace record in Output 2.2 on page 25). You can also use a partial path. A partial path consists of any part of the full path that begins immediately after a period (.) and continues to the end of the full path. For details about referencing output objects, see the discussion of specifying an output object in the documentation of the ODS SELECT statement in Chapter 3, “The ODS Statements” in The Complete Guide to the SAS Output Delivery System.

For example, to select just the output objects that contain the basic measures and the quantiles from the PROC UNIVARIATE output, use the following program.

```
/* Create HTML files. */
ods html body='select-body.htm'
   contents='select-contents.htm'
   page='select-pages.htm'
   frame='select-frame.htm';

/* Select output objects by name. */
ods select BasicMeasures Quantiles;

/* Analyze the data. */
proc univariate data=statepop mu0=3.5;
   var citypop_90 noncitypop_90;
   title;
run;

/* Close the HTML destination. */
ods html close;
```

The frame file appears in Display 2.4 on page 28. The program also creates Listing output, which is not shown. The Listing output contains the same information as the HTML body file, but it is formatted with the traditional SAS monospace font.
Display 2.4  View of the Frame File for Selected Output Objects

The contents file shows that for each variable in the analysis, PROC UNIVARIATE produces two output objects: one that contains basic measures and one that contains quantiles. All four output objects are in the body file because the ODS SELECT statement used names to identify the objects. If the ODS SELECT statement had used paths, which are unique, it could have selected output objects for the individual variables.

For more information about selecting output objects, see the documentation for the ODS SELECT statement in Chapter 3, “The ODS Statements” in The Complete Guide to the SAS Output Delivery System.

Creating an Output Data Set

The Output Delivery System also enables you to create a data set from an output object.

To create a data set, use the ODS OUTPUT statement. In this statement, you identify
- one or more output objects from which to create a data set
- the names of the data sets to create.

To create a single output data set, use this simplified form of the ODS OUTPUT statement:

```
ODS OUTPUT output-object=<SAS-data-set>;
```

Specify the output object as you do in the ODS SELECT statement: with a path, a name, a label, or a partial path. For example, to generate and print an output data set from each output object that contains the basic measures that PROC UNIVARIATE produces, use the following SAS program.

Note: This example is for the UNIX operating environment. To successfully run the example in another operating environment, you may need to change the file specifications. See Appendix 4, “Alternate ODS HTML Statements for Running Examples in Different Operating Environments,” on page 1477.
/* Turn off the generation of Listing output */
/* because you want to create a data set, not */
/* see the results. */
ods listing close;
/* Specify the data set to create. */
ods output BasicMeasures=measures;
/* When PROC UNIVARIATE runs, ODS */
/* creates a data set from the */
/* output object named BasicMeasures.*/
proc univariate data=statepop mu0=3.5;
  var citypop_90 noncitypop_90;
  title;
run;
/* Open the HTML destination for PROC PRINT. */
ods html body='measures-body.htm'
  contents='measures-contents.htm'
  frame='measures-frame.htm';
/* Print the output data set. */
proc print data=measures noobs headings=horizontal;
  title 'Output Data Set Produced from';
  title2 'PROC UNIVARIATE Basic Measures';
run;
/* Reset the destinations to their defaults. */
/* Close the HTML destination. */
ods html close;
/* Open the Listing destination. */
ods listing;
You can use the resulting data set as input to another SAS program. This program
simply prints the data set to illustrate its structure. The HTML output from PROC
PRINT appears in Display 2.5 on page 30.
Display 2.5  PROC PRINT Report of the Data Set Created by PROC UNIVARIATE and ODS

The data set contains observations for each of the variables in the VAR statement in PROC UNIVARIATE.

For more information about creating output data sets, see the discussion of the ODS OUTPUT statement in Chapter 3, “The ODS Statements,” in The Complete Guide to the SAS Output Delivery System.

Customizing Procedure Output

Many procedures that fully support ODS provide table definitions that enable you to customize each output object that the procedure produces. You do so by creating an alternate table definition for the procedure to use.

For example, the following SAS program creates a customized table definition for the BasicMeasures output object from PROC UNIVARIATE. (The trace record provides the name of the table definition that each object uses. See Output 2.2 on page 25.) In the customized version

- the measures of variability precede the measures of location
- the column headers are modified
- statistics are displayed in a bold, italic font with a 7.3 format
- a footer indicates the date that the procedure was run.

The customized HTML output object appears in Display 2.6 on page 33. The customized Listing output appears in Output 2.3 on page 33.

Note  This example is for the UNIX operating environment. To successfully run the example in another operating environment, you may need to change the file specifications. See Appendix 4, “Alternate ODS HTML Statements for Running Examples in Different Operating Environments,” on page 1477.

```sas
/* These options affect only the Listing output. */
options nodate pageno=1 linesize=80 pagesize=60;

/* This PROC TEMPLATE step creates a table definition */
/* base.univariate.Measures in the SASUSER template */
/* store. Table definitions that are provided */
/* by SAS Institute are stored in a template */
/* store in the SASHELP library. By default, ODS */
/* searches for a table definition in SASUSER before */
/* SASHELP, so when PROC UNIVARIATE calls for a */
/* table definition by this name, ODS uses the one */
/* from SASUSER. */
proc template;
   define table base.univariate.Measures;
   notes "Basic measures of location and variability";
   translate _val_ = ._ into '';

   /* The HEADER statement determines the order */
   /* in which the table definition uses the */
   /* headers, which are defined later. */
   header h1 h2 h3;
   
   /* The COLUMN statement determines the order */
   /* in which the variables appear. PROC */
   /* UNIVARIATE names the variables. */
   column VarMeasure VarValue LocMeasure LocValue;

   /* These DEFINE blocks define the headers. */
   /* They specify the text for each header. By */
   /* default, a header spans all columns, so */
   /* H1 does so. H2 spans the variables */
   /* VarMeasure and VarValue. H3 spans */
   /* LocMeasure and LocValue. */
   define h1;
      text "Basic Statistical Measures";
      spill_margin = on
      space = 1;
      end;
   define h2;
      text "Measures of Variability";
      start = VarMeasure
      end = VarValue;
      end;
   define h3;
      text "Measures of Location";
      start = LocMeasure
      end = LocValue;
      end;

   /* These DEFINE blocks specify characteristics */
   /* for each of the variables. There are two */
   /* differences between these DEFINE blocks and */
   /* the ones in the table definition in SASHELP. */
   /* These blocks use FORMAT= to specify a format */
   /* of 7.3 for LocValue and VarValue. They also */
/* use STYLE= to specify a bold, italic font */
/* for these two variables. The STYLE= option */
/* does not affect the Listing output. */
define LocMeasure;
   print_headers = off;
   glue = 2;
   space = 3;
end;

define LocValue;
   print_headers = off;
   space = 5;
   format = 7.3;
   style=data{font_style=italic font_weight=bold};
end;

define VarMeasure;
   print_headers = off;
   glue = 2;
   space = 3;
end;

define VarValue;
   print_headers = off;
   format = 7.3;
   style=data{font_style=italic font_weight=bold};
end;

/* This DEFINE block defines a footer that */
/* appears at the bottom of the table of */
/* measures. */
define footer f;
   text "Measures computed on &SysDate";
   style = FooterEmphasis;
end;

end;

/* End the PROC TEMPLATE step. */
end;

/* Begin the program that uses the */
/* customized table definition. */

/* The ODS HTML statement opens the HTML */
/* destination and identifies the files to */
/* write to. */
ods html file='statepop-body.htm' contents='statepop-contents.htm'
   page='statepop-page.htm'
   frame='statepop-frame.htm';
/* The ODS SELECT statement selects just the */
/* output object that contains the basic measures. */
ods select BasicMeasures;

/* PROC UNIVARIATE produces one object for each */
/* variable. It uses the customized table */
/* definition to format the data because the */
/* customized definition is in SASUSER. (See the */
/* explanation with the PROC TEMPLATE statement in */
/* this example. */
title;
proc univariate data=statepop mu0=3.5;
  var citypop_90 noncitypop_90;
run;

/* Close the HTML destination. */
ods html close;

Display 2.6  Customized HTML Output from PROC UNIVARIATE

Output 2.3  Customized Output from PROC UNIVARIATE
A Gallery of HTML Files Produced by Base Procedures

This section illustrates the HTML output that you can get from routing selected examples from the documentation on individual procedures through the HTML destination. Each piece of output shown was created by running the specified example with this ODS HTML statement preceding it:

```ods html body='external-file';
```

You must execute the following statement before you can view the resulting HTML files in a browser:

```ods html close;
```

**PROC TABULATE: Summarizing Information with the Universal Class Variable ALL**

The SAS program that produces this output is in Example 5 on page 1222.
**PROC FREQ: Analyzing a $2 \times 2$ Contingency Table**

The SAS program that produces this output is in Example 4 on page 579.

![Contingency Table](image1)

**PROC PRINT: Summing Numeric Variables with One BY Group**

The SAS program that produces this output is in Example 4 on page 805.

![Summing Numeric Variables](image2)
PROC REPORT: Specifying Styles for HTML Output in the PROC REPORT Statement

The SAS program that produces this output is in Example 15 on page 1006.

Customizing the Styles Used in HTML Output

What Is a Style?

A style determines the overall look of the document that uses it. Each style is a collection of style elements, each of which affects a particular part of the document. Procedures may use different style elements in different parts of their output. For example, a procedure can use one style element for column headers and another for data. Each style element is, in turn, a collection of attributes and values. The attributes determine the size, face, and weight of the type that is used, the color of the foreground and background, and other such features.

For a list of the attributes, see “What Style Attributes Can Base Procedures Specify?” on page 37.
What Styles Are Shipped with the Software?

SAS Institute ships a number of styles with the SAS System. To see a list of these styles,

1. Open the Results window.
2. Use the right mouse button to open the Templates window.
3. Expand the Sashelp.Tmplmst folder.
4. Open the Styles folder.

The Output Delivery System uses the style that is called Default unless you specify an alternative style with the STYLE= option in the ODS HTML statement (see the documentation for the ODS HTML statement in The Complete Guide to the SAS Output Delivery System).

To see the elements of a style, submit this PROC TEMPLATE step:

```plaintext
proc template;
  source style-name;
run;
```

where style-name is the path to the style from the template store (for example `styles.default` or `styles.beige`).

In most cases, if you want to alter the style of an HTML file that ODS produces, you must make a copy of the style that is used, alter that copy, and store it so that ODS will find it and use it before it finds the style that SAS Institute provides. (For information on this process, see The Complete Guide to the SAS Output Delivery System.)

How Do I Use Styles with Base Procedures?

A procedure uses one or more templates to produce output objects. Each template can specify the use of one or more style elements for various parts of the output.

However, procedures that build reports that are based on information that the user provides do not use the same templates. Two of these procedures, PROC REPORT and PROC TABULATE, provide a way for you to customize the HTML output directly from the PROC step that creates the report. Information on how to do this is provided with the syntax for these procedures.

What Style Attributes Can Base Procedures Specify?

The following list describes the style attributes that you can specify from the TABULATE and REPORT procedures. Procedures that support the Output Delivery System create HTML files and put their output in HTML tables in these files. Some of the style attributes apply to the HTML table as a whole; others apply to individual cells in the HTML table. The procedure documentation tells you which style attributes you can set from which statements in the procedure.

Note: The default value that is used for an attribute depends on the style that is in use. For information on viewing the attributes in a style, see “What Styles Are Shipped with the Software?” on page 37. The implementation of an attribute depends on the browser that you use.

Many values for style attributes are one of the following:

' string' is a quoted character string.

dimension

is a nonnegative number. The unit of measure is pixels.
color

is a string that identifies a color. A color can be

- any of the color names that are supported by SAS/GRAPH. These names include
  - a simple color name (for example, blue or VIYG)
  - a more complex color name (for example light blue)
  - a red/green/blue (RGB) value (for example, CX0023FF)
  - a hue/light/saturation (HLS) value (for example, H14E162D)
  - a gray-scale value (for example, GRAYBB).
- An RGB value of with a leading pound sign (#) rather than CX (for example, #0023FF).
- One of the colors that exists in the SAS session when the style is used:
  DMSBLUE
  DMSRED
  DMSPINK
  DMSGREEN
  DMSCYAN
  DMSYELLOW
  DMSWHITE
  DMSORANGE
  DMSBLACK
  DMSMAGENTA
  DMSGRAY
  DMSBROWN
  SYSBACK
  SYSSECB
  SYSFORE

  Note: Use these colors only if you are running SAS in the windowing environment.

- An English-like description of an HLS value. Such descriptions use a combination of words to describe the lightness, the saturation, and the hue (in that order). The words that you can use are shown in the following table:

<table>
<thead>
<tr>
<th>Lightness</th>
<th>Saturation</th>
<th>Hue</th>
</tr>
</thead>
<tbody>
<tr>
<td>black</td>
<td>gray</td>
<td>blue</td>
</tr>
<tr>
<td>very dark</td>
<td>grayish</td>
<td>purple</td>
</tr>
<tr>
<td>dark</td>
<td>moderate</td>
<td>red</td>
</tr>
<tr>
<td>medium</td>
<td>strong</td>
<td>orange</td>
</tr>
<tr>
<td>light</td>
<td>vivid</td>
<td>yellow</td>
</tr>
</tbody>
</table>
You can combine these words to form a wide variety of colors. Some examples are:

- light vivid green
- dark vivid orange
- light yellow

Note: The Output Delivery system first tries to match a color with a SAS/GRAPH color. Thus, although brown and orange are interchangeable in the table, if you use them as unmodified hues, they are different. The reason for this is that ODS treats them like SAS colors, which are mapped to different colors.

You can also specify hues that are intermediate between two neighboring colors. To do so, combine one of the following adjectives with one of its neighboring colors:

- reddish
- orangish
- brownish
- yellowish
- greenish
- bluish
- purplish

For example, you can use the following as hues:
- bluish purple (which is the same as purplish blue)
- reddish orange
- yellowish green

See also: For information on SAS/GRAPH colors, see SAS/GRAPH Software Reference.

ASIS=ON|OFF

specifies how to handle leading spaces, trailing spaces, and line breaks.

ON

prints text with leading spaces, trailing spaces, and line breaks as they are.

OFF

trims leading spaces and trailing spaces. OFF ignores line breaks.

Applies to: cells

BACKGROUND=color

specifies the color of the background.

Applies to: HTML tables or cells

BACKGROUNDIMAGE='string'

specifies an image to use as the background. Viewers that can tile the image as the background for the HTML table that the procedure creates will do so. string is the name of a GIF or JPEG file. You can use a simple file name, a complete path,
or a URL. However, the most versatile approach is to use a simple filename and to place all image files in the local directory.

**Applies to:** HTML tables or cells

**BORDERCOLOR=color**

specifies the color of the border if the border is just one color.

**Applies to:** HTML tables or cells

**BORDERCOLORDARK=color**

specifies the darker color to use in a border that uses two colors to create a three-dimensional effect.

**Applies to:** HTML tables or cells

**BORDERCOLORLIGHT=color**

specifies the lighter color to use in a border that uses two colors to create a three-dimensional effect.

**Applies to:** HTML tables or cells

**BORDERWIDTH=dimension**

specifies the width of the border of the HTML table.

**Applies to:** HTML tables or cells

**Tip:** Typically, when BORDERWIDTH=0, the browser sets RULES=NONE (see the discussion of RULES= on page 44) and FRAME=VOID (see the discussion of FRAME= on page 42).

**CELLHEIGHT=dimension|integer%**

specifies the height of the cell. If you specify a percent, it represents a percentage of the height of the table. A row of cells will have the height of the highest cell in the row.

**Tip:** HTML automatically sets cell height appropriately. You should seldom need to specify this attribute.

**Applies to:** cells

**CELLPADDING=dimension|integer%**

specifies the amount of white space on each of the four sides of the text in a cell.

**Applies to:** HTML tables

**CELLSPACING=dimension**

specifies the thickness of the spacing between cells.

**Applies to:** HTML tables

**Interaction:** If BORDERWIDTH= is nonzero, and if the background color of the cells contrasts with the background color of the table, the cell spacing appears as a rule that is the same color as the table’s background.

**CELLWIDTH=dimension|integer%**

specifies the width of the cell. If you specify a percent, it represents a percentage of the width of the table. A column of cells will have the width of the widest cell in the column.

**Applies to:** cells

**Tip:** HTML automatically sets cell width appropriately. You should seldom need to specify this attribute.
FLYOVER='string'
specifies the text to show in a tool tip for the cell.

**Applies to:** cells

FONT_FACE='string-1', 'string-n'>
specifies the font face to use. If you supply more than one string, the browser uses the first one that is installed on your system.

You cannot be sure what fonts are available to someone who is viewing your HTML output in a browser. Most devices support

- times
- courier
- arial, helvetica

**Applies to:** HTML tables or cells

FONT_SIZE=size
specifies the size of the font. The interpretation of the value of size depends on the browser.

**Applies to:** HTML tables or cells

**Range:** 1 to 7

FONT_STYLE=ITALIC | ROMAN | SLANT
specifies the style of the font. In many cases, italic and slant map to the same font.

**Applies to:** HTML tables or cells

FONT_WEIGHT=weight
specifies the font weight. weight can be any of the following:

- MEDIUM
- BOLD
- DEMI_BOLD
- EXTRA_BOLD
- LIGHT
- DEMI_LIGHT
- EXTRA_LIGHT
- BLACK

**Applies to:** HTML tables or cells

**Restriction:** You cannot be sure what font weights are available to someone who is viewing your HTML output in a browser. Most devices support only MEDIUM and BOLD, and possibly LIGHT.

FONT_WIDTH=relative-width
specifies the font width compared to the width of the usual design. relative-width can be any of the following:

- NORMAL
- COMPRESSED
- EXTRA_COMPRESSED
- NARROW
- WIDE
- EXPANDED
Applies to: HTML tables or cells
Restriction: Most fonts do not honor these values.

\texttt{FOREGROUND=color}

specifies the color of the foreground.

\textbf{Applies to:} HTML tables or cells

\texttt{FRAME=frame-type}

specifies the type of frame to use on an HTML table. The following table shows the possible values of frame-type and their meanings.

<table>
<thead>
<tr>
<th>This value of frame-type</th>
<th>Creates this kind of frame around the table</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABOVE</td>
<td>a border at the top</td>
</tr>
<tr>
<td>BELOW</td>
<td>a border at the bottom</td>
</tr>
<tr>
<td>BOX</td>
<td>borders at the top, bottom, and both sides</td>
</tr>
<tr>
<td>HSIDES</td>
<td>borders at the top and bottom</td>
</tr>
<tr>
<td>LHS</td>
<td>a border at the left side</td>
</tr>
<tr>
<td>RHS</td>
<td>a border at the right side</td>
</tr>
<tr>
<td>VOID</td>
<td>no borders</td>
</tr>
<tr>
<td>VSIDES</td>
<td>borders at the left and right sides</td>
</tr>
</tbody>
</table>

\textbf{Applies to:} HTML tables

\texttt{HTMLCLASS='string'}

specifies the name of the stylesheet class to use for the table or cell.

\textbf{Applies to:} HTML tables and cells

\texttt{HTMLID='string'}

specifies an id for the table or cell. The id is for use by a JavaScript.

\textbf{Applies to:} HTML tables and cells

\texttt{HTMLSTYLE='string'}

specifies individual attributes and values for the table or cell.

\textbf{Applies to:} HTML tables and cells

\texttt{JUST=justification}

specifies justification, where justification can be

\texttt{CENTER}

specifies center justification.

Alias: C

\textbf{Applies to:} HTML tables

\texttt{LEFT}

specifies left justification.

Alias: L

\textbf{Applies to:} cells

\texttt{RIGHT}

specifies right justification.
Alias: R
Applies to: cells
Restriction: Not all contexts support RIGHT. If RIGHT is not supported, it is interpreted as CENTER.

NOBREAKSPACE=ON | OFF
specifies how to handle space characters.

ON
does not break a line at a space character.

OFF
breaks a line at a space character if appropriate.

Applies to: cells

OUTPUTWIDTH=dimension | integer%
specifies the width of the HTML table. If you specify a percent, it represents a percentage of the width of the browser window.

Applies to: HTML tables
Tip: Use OUTPUTWIDTH=100% to make the HTML table as wide as the window that it is open in.

POSTHTML='string'
specifies the HTML code to place after the HTML table or cell.

Applies to: HTML tables or cells

POSTIMAGE='string'
specifies an image to place after the HTML table or cell. string is the name of a GIF or JPEG file. You can use a simple filename, a complete path, or a URL. However, the most versatile approach is to use a simple filename and to place all image files in the local directory.

Applies to: HTML tables or cells

POSTTEXT='string'
specifies text to place after the cell or HTML table.

Applies to: HTML tables or cells

PREHTML='string'
specifies the HTML code to place before the HTML table or cell.

Applies to: HTML tables or cells

PREIMAGE='string'
specifies an image to place before the HTML table or cell. string is the name of a GIF or JPEG file. You can use a simple filename, a complete path, or a URL. However, the most versatile approach is to use a simple filename and to place all image files in the local directory.

Applies to: HTML tables or cells

PRETEXT='string'
specifies text to place before the cell or HTML table.

Applies to: HTML tables or cells
PROTECTSPECIALCHARACTERS=ON | OFF | AUTO
determines how less-than signs (<), greater-than signs (>), and ampersands (&) are interpreted. In HTML, these characters indicate the beginning of a markup tag, the end of a markup tag, and the beginning of the name of a file or character entity.

ON
interprets special characters as the characters themselves. That is, when ON is in effect the characters are protected before they are passed to the HTML destination so that HTML does not interpret them as part of the markup language. Using ON enables you to show HTML markup in your document.

OFF
interprets special characters as HTML code. That is, when OFF is in effect, the characters are passed to the HTML destination without any protection so that HTML interprets them as part of the markup language.

AUTO
interprets any string that starts with a < and ends with a > as HTML (ignoring spaces that immediately follow the <, spaces that immediately precede the >, and spaces at the beginning and end of the string). In any other string, AUTO protects the special characters from their HTML meaning.

**Applies to:** HTML tables or cells

RULES=rule-type
specifies the types of rules to use in an HTML table. The following table shows the possible values of rule and their meanings.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Creates rules in these locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>between all rows and columns</td>
</tr>
<tr>
<td>COLS</td>
<td>between all columns</td>
</tr>
<tr>
<td>GROUP</td>
<td>between the table header and the table and between the table and the table footer, if there is one</td>
</tr>
<tr>
<td>NONE</td>
<td>no rules anywhere</td>
</tr>
<tr>
<td>ROWS</td>
<td>between all rows</td>
</tr>
</tbody>
</table>

**Applies to:** HTML tables

VJUST='justification'
specifies vertical justification, where justification can be

TOP
specifies top justification.
Alias: T

BOTTOM
specifies bottom justification.
Alias: B

MIDDLE
specifies center justification.
RUN-Group Processing

RUN-group processing enables you to submit a PROC step with a RUN statement without ending the procedure. You can continue to use the procedure without issuing another PROC statement. To end the procedure, use a RUN CANCEL or a QUIT statement. Several base SAS procedures support RUN-group processing:
Creating Titles That Contain BY-Group Information

BY-group processing uses a BY statement to process observations that are ordered, grouped, or indexed according to the values of one or more variables. By default, when you use BY-group processing in a procedure step, a BY line identifies each group. This section explains how to create titles that serve as customized BY lines.

Suppressing the Default BY Line

When you insert BY-group processing information into a title, you usually want to eliminate the default BY line. To suppress it, use the SAS system option NOBYLINE.

Note: You must use the NOBYLINE option if you insert BY-group information into titles for the following base SAS procedures:

MEANS
PRINT
STANDARD
SUMMARY.

If you use the BY statement with the NOBYLINE option, these procedures always start a new page for each BY group. This behavior prevents multiple BY groups from appearing on a single page and ensures that the information in the titles matches the report on the pages.

Inserting BY-Group Information into a Title

The general form for inserting BY-group information into a title is

#BY-specification<suffix>

BY-specification

is one of the following:

BYVAL n | BYVAL(BY-variable)

places the value of the specified BY variable in the title. You specify the BY variable with one of the following:

n
is the nth BY variable in the BY statement.

BY-variable

is the name of the BY variable whose value you want to insert in the title.
BYVARn | BYVAR(BY-variable)

places the label or the name (if no label exists) of the specified BY variable in
the title. You designate the BY variable with one of the following:

n
is the nth BY variable in the BY statement.

BY-variable
is the name of the BY variable whose name you want to insert in the
title.

BYLINE
inserts the complete default BY line into the title.

suffix
supplies text to place immediately after the BY-group information that you insert
in the title. No space appears between the BY-group information and the suffix.

Example: Inserting a Value from Each BY Variable into the Title

This example
1 creates a data set, GROC, that contains data for stores from four regions. Each
store has four departments. This data set is created in a DATA step “GROC” on
page 1444.
2 sorts the data by Region and Department.
3 uses the SAS system option NOBYLINE to suppress the BY line that normally
appears in output that is produced with BY-group processing.
4 uses PROC CHART to chart sales by Region and Department. In the first TITLE
statement, #BYVAL2 inserts the value of the second BY variable, Department, into
the title. In the second TITLE statement, #BYVAL(Region) inserts the value of
Region into the title. The first period after Region indicates that a suffix follows.
The second period is the suffix.
5 uses the SAS system option BYLINE to return to the creation of the default BY
line with BY-group processing.

data groc;
   input Region $9. Manager $ Department $ Sales;
datalines;
Southeast Hayes Paper 250
Southeast Hayes Produce 100
Southeast Hayes Canned 120
Southeast Hayes Meat 80
... more lines of data...
Northeast Fuller Paper 200
Northeast Fuller Produce 300
Northeast Fuller Canned 420
Northeast Fuller Meat 125
;

proc sort data=groc;
   by region department;
run;
options nobyline nodate pageno=1
   linesize=64 pagesize=20;
proc chart data=groc;
   by region department;
vbar manager / type=sum sumvar=sales;
title1 'This chart shows #byval2 sales';
title2 'in the #byval(region)...';
run;
options byline;

This partial output shows two BY groups with customized BY lines:

<table>
<thead>
<tr>
<th>Sales Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>400 + ***** *****</td>
</tr>
<tr>
<td>300 + ***** *****</td>
</tr>
<tr>
<td>200 + ***** ***** *****</td>
</tr>
<tr>
<td>100 + ***** ***** ***** *****</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Aikmann</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>Manager</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sales Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>75 + ***** *****</td>
</tr>
<tr>
<td>60 + ***** *****</td>
</tr>
<tr>
<td>45 + ***** *****</td>
</tr>
<tr>
<td>30 + ***** ***** *****</td>
</tr>
<tr>
<td>15 + ***** ***** ***** *****</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Aikmann</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>Manager</td>
</tr>
</tbody>
</table>

**Example: Inserting the Name of a BY Variable into a Title**

This example inserts the name of a BY variable and the value of a BY variable into the title. The program
1 uses the SAS system option NOBYLINE to suppress the BY line that normally appears in output that is produced with BY-group processing.
2 uses PROC CHART to chart sales by Region. In the first TITLE statement, \#BYVAR(Region) inserts the name of the variable Region into the title. (If Region had a label, \#BYVAR would use the label instead of the name.) The suffix al is appended to the label. In the second TITLE statement, \#BYVAL1 inserts the value of the first BY variable, Region, into the title.

3 uses the SAS system option BYLINE to return to the creation of the default BY line with BY-group processing.

```
options nobyline nodate pageno=1
   linesize=64 pagesize=20;
proc chart data=groc;
   by region;
   vbar manager / type=mean sumvar=sales;
   title1 '#byvar(region).al Analysis';
   title2 'for the #byval1';
run;
options byline;
```

This partial output shows one BY group with a customized BY line:

```
Regional Analysis 1
for the Northwest

Sales Mean
300 + *****
| *****
200 + ***** *****
| ***** *****
100 + ***** ***** *****
| ***** ***** *****
--------------------------------------------
Aikmann Duncan Jeffreys
Manager
```

**Example: Inserting the Complete BY Line into a Title**

This example inserts the complete BY line into the title. The program

1 uses the SAS system option NOBYLINE to suppress the BY line that normally appears in output that is produced with BY-group processing.

2 uses PROC CHART to chart sales by Region and Department. In the TITLE statement, \#BYLINE inserts the complete BY line into the title.

3 uses the SAS system option BYLINE to return to the creation of the default BY line with BY-group processing.

```
options nobyline nodate pageno=1
   linesize=64 pagesize=20;
```
proc chart data=groc;
  by region department;
  vbar manager / type=sum sumvar=sales;
  title 'Information for #byline';
run;
options byline;

This partial output shows two BY groups with customized BY lines:

| Information for Region=Northwest Department=Canned 1 |
| Sales Sum |
| 400 + ***** ***** |
| | ***** ***** |
| 300 + ***** ***** |
| | ***** ***** |
| 200 + ***** ***** ***** |
| | ***** ***** ***** |
| 100 + ***** ***** ***** |
| | ***** ***** ***** |
| -------------------------------------------- Aikmann Duncan Jeffreys |

| Information for Region=Northwest Department=Meat 2 |
| Sales Sum |
| 75 + ***** ***** |
| | ***** ***** |
| 60 + ***** ***** |
| | ***** ***** |
| 45 + ***** ***** |
| | ***** ***** |
| 30 + ***** ***** ***** |
| | ***** ***** ***** |
| 15 + ***** ***** ***** |
| | ***** ***** ***** |
| -------------------------------------------- Aikmann Duncan Jeffreys |

Error Processing of BY-Group Specifications

The SAS System does not issue error or warning messages for incorrect #BYVAL, #BYVAR, or #BYLINE specifications. Instead, the text of the item simply becomes part of the title.

Shortcuts for Specifying Lists of Variable Names

Several statements in procedures allow multiple variable names. You can use these shortcut notations instead of specifying each variable name:
### Notation | Meaning
--- | ---
x1-xn | specifies variables X1 through Xn. The numbers must be consecutive.
x: | specifies all variables that begin with the letter X.
x--a | specifies all variables between X and A, inclusive. This notation uses the position of the variables in the data set.
x-numeric-a | specifies all numeric variables between X and A, inclusive. This notation uses the position of the variables in the data set.
x-character-a | specifies all character variables between X and A, inclusive. This notation uses the position of the variables in the data set.
_numeric_ | specifies all numeric variables.
_character_ | specifies all character variables.
_all_ | specifies all variables.

**Note:** You cannot use shortcuts to list variable names in the INDEX CREATE statement in PROC DATASETS.


### Formatted Values
Typically, when you print or group variable values, base SAS procedures use the formatted values. This section contains examples of how base procedures use formatted values.

#### Example: Printing the Formatted Values for a Data Set
The following example prints the formatted values of the data set PROCLIB.PAYROLL. (A DATA step "PROCLIB.PAYROLL" on page 1451 creates this data set.) In PROCLIB.PAYROLL, the variable Jobcode indicates the job and level of the employee. For example, TA1 indicates that the employee is at the beginning level for a ticket agent.

```sas
libname proclib 'SAS-data-library';

options nodate pageno=1
   linesize=64 pagesize=40;
```
proc print data=proclib.payroll(obs=10) noobs;
title 'PROCLIB.PAYROLL';
title2 'First 10 Observations Only';
run;

This is a partial printing of PROCLIB.PAYROLL:

<table>
<thead>
<tr>
<th>Id</th>
<th>Number</th>
<th>Sex</th>
<th>Jobcode</th>
<th>Salary</th>
<th>Birth</th>
<th>Hired</th>
</tr>
</thead>
<tbody>
<tr>
<td>1919</td>
<td>M</td>
<td>TA2</td>
<td>34376</td>
<td>12SEP60</td>
<td>04JUN87</td>
<td></td>
</tr>
<tr>
<td>1653</td>
<td>F</td>
<td>ME2</td>
<td>35108</td>
<td>15OCT64</td>
<td>09AUG90</td>
<td></td>
</tr>
<tr>
<td>1400</td>
<td>M</td>
<td>ME1</td>
<td>29769</td>
<td>05NOV67</td>
<td>16OCT90</td>
<td></td>
</tr>
<tr>
<td>1350</td>
<td>F</td>
<td>FA3</td>
<td>32886</td>
<td>31AUG65</td>
<td>29JUL90</td>
<td></td>
</tr>
<tr>
<td>1401</td>
<td>M</td>
<td>TA3</td>
<td>38822</td>
<td>13DEC50</td>
<td>17NOV85</td>
<td></td>
</tr>
<tr>
<td>1499</td>
<td>M</td>
<td>ME3</td>
<td>43025</td>
<td>26APR54</td>
<td>07JUN80</td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td>M</td>
<td>SCP</td>
<td>18723</td>
<td>06JUN62</td>
<td>01OCT90</td>
<td></td>
</tr>
<tr>
<td>1333</td>
<td>M</td>
<td>PT2</td>
<td>88606</td>
<td>30MAR61</td>
<td>10FEB81</td>
<td></td>
</tr>
<tr>
<td>1402</td>
<td>M</td>
<td>TA2</td>
<td>32615</td>
<td>17JAN63</td>
<td>02DEC90</td>
<td></td>
</tr>
<tr>
<td>1479</td>
<td>F</td>
<td>TA3</td>
<td>38785</td>
<td>22DEC68</td>
<td>05OCT89</td>
<td></td>
</tr>
</tbody>
</table>

The following PROC FORMAT step creates the format $JOBMT., which assigns descriptive names for each job:

proc format;
  value $jobfmt
    'FA1'='Flight Attendant Trainee'
    'FA2'='Junior Flight Attendant'
    'FA3'='Senior Flight Attendant'
    'ME1'='Mechanic Trainee'
    'ME2'='Junior Mechanic'
    'ME3'='Senior Mechanic'
    'PT1'='Pilot Trainee'
    'PT2'='Junior Pilot'
    'PT3'='Senior Pilot'
    'TA1'='Ticket Agent Trainee'
    'TA2'='Junior Ticket Agent'
    'TA3'='Senior Ticket Agent'
    'NA1'='Junior Navigator'
    'NA2'='Senior Navigator'
    'BCK'='Baggage Checker'
    'SCP'='Skycap';
run;

The FORMAT statement in this PROC MEANS step temporarily associates the $JOBMT. format with the variable J obcode:

options nodate pageno=1 linesize=64 pagesize=60;
proc means data=proclib.payroll mean max;
class jobcode;
var salary;
format jobcode $jobfmt.;
title 'Summary Statistics for';
title2 'Each Job Code';
run;

PROC MEANS produces this output, which uses the $JOBFMT. format:

<table>
<thead>
<tr>
<th>Jobcode</th>
<th>N</th>
<th>Obs</th>
<th>Mean</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baggage Checker</td>
<td>9</td>
<td>25794.22</td>
<td>26896.00</td>
<td></td>
</tr>
<tr>
<td>Flight Attendant Trainee</td>
<td>11</td>
<td>23039.36</td>
<td>23979.00</td>
<td></td>
</tr>
<tr>
<td>Junior Flight Attendant</td>
<td>16</td>
<td>27986.88</td>
<td>28978.00</td>
<td></td>
</tr>
<tr>
<td>Senior Flight Attendant</td>
<td>7</td>
<td>32933.86</td>
<td>33419.00</td>
<td></td>
</tr>
<tr>
<td>Mechanic Trainee</td>
<td>8</td>
<td>28500.25</td>
<td>29769.00</td>
<td></td>
</tr>
<tr>
<td>Junior Mechanic</td>
<td>14</td>
<td>35576.86</td>
<td>36925.00</td>
<td></td>
</tr>
<tr>
<td>Senior Mechanic</td>
<td>7</td>
<td>42410.71</td>
<td>43900.00</td>
<td></td>
</tr>
<tr>
<td>Junior Navigator</td>
<td>5</td>
<td>42032.20</td>
<td>43433.00</td>
<td></td>
</tr>
<tr>
<td>Senior Navigator</td>
<td>3</td>
<td>52383.00</td>
<td>53798.00</td>
<td></td>
</tr>
<tr>
<td>Pilot Trainee</td>
<td>8</td>
<td>67908.00</td>
<td>71349.00</td>
<td></td>
</tr>
<tr>
<td>Junior Pilot</td>
<td>10</td>
<td>87925.20</td>
<td>91908.00</td>
<td></td>
</tr>
<tr>
<td>Senior Pilot</td>
<td>2</td>
<td>10504.50</td>
<td>11379.00</td>
<td></td>
</tr>
<tr>
<td>Skycap</td>
<td>7</td>
<td>18308.86</td>
<td>18833.00</td>
<td></td>
</tr>
<tr>
<td>Ticket Agent Trainee</td>
<td>9</td>
<td>27721.33</td>
<td>28880.00</td>
<td></td>
</tr>
<tr>
<td>Junior Ticket Agent</td>
<td>20</td>
<td>33574.95</td>
<td>34803.00</td>
<td></td>
</tr>
<tr>
<td>Senior Ticket Agent</td>
<td>12</td>
<td>39679.58</td>
<td>40899.00</td>
<td></td>
</tr>
</tbody>
</table>

Note: Because formats are character strings, formats for numeric variables are ignored when the values of the numeric variables are needed for mathematical calculations.

Example: Grouping or Classifying Formatted Data

If you use a formatted variable to group or classify data, the procedure uses the formatted values. The following example creates and assigns a format, $CODEFMT., that groups the levels of each job code into one category. PROC MEANS calculates statistics based on the groupings of the $CODEFMT. format.
proc format;
  value $codefmt
    'FA1','FA2','FA3'='Flight Attendant'
    'ME1','ME2','ME3'='Mechanic'
    'PT1','PT2','PT3'='Pilot'
    'TA1','TA2','TA3'='Ticket Agent'
    'NA1','NA2'='Navigator'
    'BCK'='Baggage Checker'
    'SCP'='Skycap';
run;

options nodate pageno=1
  linesize=64 pagesize=40;
proc means data=proclib.payroll mean max;
  class jobcode;
  var salary;
  format jobcode $codefmt.;
  title 'Summary Statistics for Job Codes';
  title2 '(Using a Format that Groups the Job Codes)';
run;

PROC MEANS produces this output:

<table>
<thead>
<tr>
<th>Jobcode</th>
<th>N</th>
<th>Obs</th>
<th>Mean</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baggage Checker</td>
<td>9</td>
<td>25794.22</td>
<td>26896.00</td>
<td></td>
</tr>
<tr>
<td>Flight Attendant</td>
<td>34</td>
<td>27404.71</td>
<td>33419.00</td>
<td></td>
</tr>
<tr>
<td>Mechanic</td>
<td>29</td>
<td>35274.24</td>
<td>43900.00</td>
<td></td>
</tr>
<tr>
<td>Navigator</td>
<td>8</td>
<td>45913.75</td>
<td>53798.00</td>
<td></td>
</tr>
<tr>
<td>Pilot</td>
<td>20</td>
<td>72176.25</td>
<td>91908.00</td>
<td></td>
</tr>
<tr>
<td>Skycap</td>
<td>7</td>
<td>18308.86</td>
<td>18833.00</td>
<td></td>
</tr>
<tr>
<td>Ticket Agent</td>
<td>41</td>
<td>34076.73</td>
<td>40899.00</td>
<td></td>
</tr>
</tbody>
</table>

Example: Temporarily Associating a Format with a Variable

If you want to associate a format with a variable temporarily, you can use the FORMAT statement. For example, the following PROC PRINT step associates the DOLLAR8. format with the variable Salary for the duration of this PROC PRINT step only:

```latex
options nodate pageno=1
  linesize=64 pagesize=40;
proc print data=proclib.payroll(obs=10)
  noobs;
  format salary dollar8.;
```
title 'Temporarily Associating a Format';
  title2 'with the Variable Salary';
run;

PROC PRINT produces this output:

Temporarily Associating a Format 1
with the Variable Salary

<table>
<thead>
<tr>
<th>Id</th>
<th>Number</th>
<th>Sex</th>
<th>Jobcode</th>
<th>Salary</th>
<th>Birth</th>
<th>Hired</th>
</tr>
</thead>
<tbody>
<tr>
<td>1919</td>
<td>M</td>
<td>TA2</td>
<td>$34,376</td>
<td>12SEP60</td>
<td>04JUN87</td>
<td></td>
</tr>
<tr>
<td>1653</td>
<td>F</td>
<td>ME2</td>
<td>$35,108</td>
<td>15OCT64</td>
<td>09AUG90</td>
<td></td>
</tr>
<tr>
<td>1400</td>
<td>M</td>
<td>ME1</td>
<td>$29,769</td>
<td>05NOV97</td>
<td>16OCT90</td>
<td></td>
</tr>
<tr>
<td>1350</td>
<td>F</td>
<td>FA3</td>
<td>$32,886</td>
<td>31AUG65</td>
<td>29JUL90</td>
<td></td>
</tr>
<tr>
<td>1401</td>
<td>M</td>
<td>TA3</td>
<td>$38,822</td>
<td>13DEC50</td>
<td>17NOV85</td>
<td></td>
</tr>
<tr>
<td>1499</td>
<td>M</td>
<td>ME3</td>
<td>$43,025</td>
<td>26APR54</td>
<td>07JUN80</td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td>M</td>
<td>SCP</td>
<td>$18,723</td>
<td>06JUN62</td>
<td>01OCT90</td>
<td></td>
</tr>
<tr>
<td>1333</td>
<td>M</td>
<td>PT2</td>
<td>$88,606</td>
<td>30MAR61</td>
<td>10FEB81</td>
<td></td>
</tr>
<tr>
<td>1402</td>
<td>M</td>
<td>TA2</td>
<td>$32,615</td>
<td>17JAN63</td>
<td>02DEC90</td>
<td></td>
</tr>
<tr>
<td>1479</td>
<td>F</td>
<td>TA3</td>
<td>$38,785</td>
<td>22DEC68</td>
<td>05OCT89</td>
<td></td>
</tr>
</tbody>
</table>

Example: Temporarily Dissociating a Format from a Variable

If a variable has a permanent format that you do not want a procedure to use, temporarily dissociate the format from the variable using a FORMAT statement.

In this example, the FORMAT statement in the DATA step permanently associates the $YRFMT. variable with the variable Year. Thus, when you use the variable in a PROC step, the procedure uses the formatted values. The PROC MEANS step, however, contains a FORMAT statement that dissociates the $YRFMT. format from Year for this PROC MEANS step only. PROC MEANS uses the stored value for Year in the output.

```
proc format;
  value $yrfmt '1'='Freshman'
         '2'='Sophomore'
         '3'='Junior'
         '4'='Senior';
run;
```
data debate;
  input Name $ Gender $ Year $ GPA @@;
  format year $yrfmt.;
  datalines;
  Capiccio m 1 3.598 Tucker m 1 3.901
  Bagwell f 2 3.722 Berry m 2 3.198
  Metcalf m 2 3.342 Gold f 3 3.609
  Gray f 3 3.177 Syme f 3 3.883
  Baglione f 4 4.000 Carr m 4 3.750
  Hall m 4 3.574 Lewis m 4 3.421
;
options nodate pageno=1
  linesize=64 pagesize=40;
proc means data=debate mean maxdec=2;
  class year;
  format year;
  title 'Average GPA';
run;

PROC MEANS produces this output, which does not use the YRFMT. format:

```
The MEANS Procedure
Analysis Variable : GPA

  N  Obs  Mean
Year                  -------------------
  1  2   3.75
  2  3   3.42
  3  3   3.56
  4  4   3.69
```

Formats and BY-Group Processing

When a procedure processes a data set, it checks to see if a format is assigned to the
BY variable. If so, the procedure adds observations to the current BY groups until the
formatted value changes. If nonconsecutive internal values of the BY variable(s) have
the same formatted value, the values are grouped into different BY groups. This results
in two BY groups with the same formatted value. Further, if different and consecutive
internal values of the BY variable(s) have the same formatted value, they are included
in the same BY group.

Formats and Error Checking

If SAS cannot find a format, it stops processing and prints an error message in the
SAS log. You can suppress this behavior with the SAS system option NOFMTERR.
When you use NOFMTERR, and SAS cannot find the format, SAS uses a default format
and continues to process. Typically, for the default, SAS uses the BESTw. format for
numeric variables and the $w. format for character variables.
Note: To ensure that SAS can find user-written formats, use the SAS system option FMTSEARCH=. How to store formats is described in “Storing Informats and Formats” on page 455.

Processing All the Data Sets in a Library

You can use the SAS Macro Facility to run the same procedure on every data set in a library. The macro facility is part of base SAS software.

Example 9 on page 819 shows how to print all the data sets in a library. You can use the same macro definition to perform any procedure on all the data sets in a library. Simply replace the PROC PRINT piece of the program with the appropriate procedure code.

Operating Environment-Specific Procedures

Several base SAS procedures are specific to one operating environment or one release. Appendix 2, “Operating Environment-Specific Procedures,” on page 1431 contains a table with additional information. These procedures are described in more detail in the SAS documentation for operating environments.

Statistic Descriptions

Table 2.1 on page 57 identifies common descriptive statistics that are available in several base procedures. See “Keywords and Formulas” on page 1397 for more detailed information about available statistics and theoretical information.

Table 2.1 Common Descriptive Statistics That Base Procedures Calculate

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Description</th>
<th>Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>confidence intervals</td>
<td></td>
<td>FREQ, MEANS, UNIVARIATE</td>
</tr>
<tr>
<td>CSS</td>
<td>corrected sum of squares</td>
<td>CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE</td>
</tr>
<tr>
<td>CV</td>
<td>coefficient of variation</td>
<td>MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE</td>
</tr>
<tr>
<td>goodness-of-fit tests</td>
<td></td>
<td>FREQ, UNIVARIATE</td>
</tr>
<tr>
<td>KURTOSIS</td>
<td>kurtosis</td>
<td>MEANS/SUMMARY, UNIVARIATE</td>
</tr>
<tr>
<td>MAX</td>
<td>largest (maximum) value</td>
<td>CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE</td>
</tr>
<tr>
<td>MEAN</td>
<td>mean</td>
<td>CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE</td>
</tr>
<tr>
<td>MEDIAN</td>
<td>median (50th percentile)</td>
<td>CORR (for nonparametric correlation measures), MEANS/SUMMARY, TABULATE, UNIVARIATE</td>
</tr>
<tr>
<td>MIN</td>
<td>smallest (minimum) value</td>
<td>CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE</td>
</tr>
</tbody>
</table>
### Computational Requirements for Statistics

The following requirements are computational requirements for the statistics that are listed in Table 2.1 on page 57. They do not describe recommended sample sizes.

- **N** and **NMISS** do not require any nonmissing observations.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Description</th>
<th>Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE</td>
<td>most frequent value (if not unique, the smallest mode is used)</td>
<td>UNIVARIATE</td>
</tr>
<tr>
<td>N</td>
<td>number of observations on which calculations are based</td>
<td>CORR, FREQ, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE</td>
</tr>
<tr>
<td>NMISS</td>
<td>number of missing values</td>
<td>FREQ, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE</td>
</tr>
<tr>
<td>NOBS</td>
<td>number of observations</td>
<td>MEANS/SUMMARY, UNIVARIATE</td>
</tr>
<tr>
<td>PCTN</td>
<td>the percentage of a cell or row frequency to a total frequency</td>
<td>REPORT, TABULATE</td>
</tr>
<tr>
<td>PCTSUM</td>
<td>the percentage of a cell or row sum to a total sum</td>
<td>REPORT, TABULATE</td>
</tr>
<tr>
<td>Pearson correlation</td>
<td></td>
<td>CORR</td>
</tr>
<tr>
<td>percentiles</td>
<td></td>
<td>FREQ, MEANS/SUMMARY, TABULATE, UNIVARIATE</td>
</tr>
<tr>
<td>RANGE</td>
<td>range</td>
<td>CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE</td>
</tr>
<tr>
<td>robust statistics</td>
<td>trimmed means, Winsorized means</td>
<td>UNIVARIATE</td>
</tr>
<tr>
<td>SKEWNESS</td>
<td>skewness</td>
<td>MEANS/SUMMARY, UNIVARIATE</td>
</tr>
<tr>
<td>Spearman correlation</td>
<td></td>
<td>CORR</td>
</tr>
<tr>
<td>STD</td>
<td>standard deviation</td>
<td>CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE</td>
</tr>
<tr>
<td>STDERR</td>
<td>the standard error of the mean</td>
<td>MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE</td>
</tr>
<tr>
<td>SUM</td>
<td>sum</td>
<td>CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE</td>
</tr>
<tr>
<td>SUMWGT</td>
<td>sum of weights</td>
<td>CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE</td>
</tr>
<tr>
<td>tests of location</td>
<td></td>
<td>UNIVARIATE</td>
</tr>
<tr>
<td>USS</td>
<td>uncorrected sum of squares</td>
<td>CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE</td>
</tr>
<tr>
<td>VAR</td>
<td>variance</td>
<td>CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE</td>
</tr>
</tbody>
</table>
- SUM, MEAN, MAX, MIN, RANGE, USS, and CSS require at least one nonmissing observation.
- VAR, STD, STDERR, and CV require at least two observations.
- CV requires that MEAN is not equal to zero.

Statistics are reported as missing if they cannot be computed.