

## CHAPTER

## 2

# Fundamental Concepts for Using Base SAS Procedures

<i>Language Concepts</i>	16
<i>Temporary and Permanent SAS Data Sets</i>	16
<i>USER Data Library</i>	16
<i>SAS System Options</i>	17
<i>Data Set Options</i>	17
<i>Global Statements</i>	18
<i>Procedure Concepts</i>	18
<i>Input Data Sets</i>	18
<i>Output Delivery System</i>	19
<i>Creating Listing Output</i>	20
<i>Creating Printer Output</i>	20
<i>Creating HTML Output</i>	21
<i>Identifying Output Objects</i>	23
<i>Selecting Output Objects to Send to ODS Destinations</i>	26
<i>Creating an Output Data Set</i>	28
<i>Storing Links in the Results Folder</i>	30
<i>Customizing Procedure Output</i>	31
<i>A Gallery of HTML and Printer Output Produced by Base Procedures</i>	36
<i>PROC TABULATE: Summarizing Information with the Universal Class Variable ALL</i>	37
<i>PROC FREQ: Analyzing a 2×2 Contingency Table</i>	37
<i>PROC PRINT: Summing Numeric Variables with One BY Group</i>	40
<i>PROC REPORT: Specifying Style Elements for HTML Output in the PROC REPORT Statement</i>	41
<i>Customizing the Style Definition That ODS Uses</i>	42
<i>What Is a Style Definition?</i>	42
<i>What Style Definitions Are Shipped with the Software?</i>	43
<i>How Do I Use Styles with Base Procedures?</i>	43
<i>What Style Attributes Can Base Procedures Specify?</i>	43
<i>RUN-Group Processing</i>	53
<i>Creating Titles That Contain BY-Group Information</i>	54
<i>Suppressing the Default BY Line</i>	54
<i>Inserting BY-Group Information into a Title</i>	54
<i>Example: Inserting a Value from Each BY Variable into the Title</i>	55
<i>Example: Inserting the Name of a BY Variable into a Title</i>	56
<i>Example: Inserting the Complete BY Line into a Title</i>	57
<i>Error Processing of BY-Group Specifications</i>	58
<i>Shortcuts for Specifying Lists of Variable Names</i>	58
<i>Formatted Values</i>	59
<i>Example: Printing the Formatted Values for a Data Set</i>	59
<i>Example: Grouping or Classifying Formatted Data</i>	61
<i>Example: Temporarily Associating a Format with a Variable</i>	62

<i>Example: Temporarily Dissociating a Format from a Variable</i>	63
<i>Formats and BY-Group Processing</i>	64
<i>Formats and Error Checking</i>	64
<i>Processing All the Data Sets in a Library</i>	64
<i>Operating Environment-Specific Procedures</i>	64
<i>Statistic Descriptions</i>	65
<i>Computational Requirements for Statistics</i>	66

---

## 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;
```

```
proc print data=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=
DLDMGACTION=	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:

<i>comment</i>	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 19 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 became 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, output that is formatted for a high-resolution printer, and output that is formatted in HyperText 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, traditional Listing output, or Printer 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*.

*Note:* The examples in this section use filenames that may not be valid in all operating environments. To successfully run the example in your operating environment, you may need to change the file specifications. See Appendix 1, “Alternate ODS HTML Statements for Running Examples in Different Operating Environments,” in *The Complete Guide to the SAS Output Delivery System*. △

## Creating Listing Output

You do not need to change your SAS programs to create Listing output. By default, the Listing destination is open. Unless you specifically close the Listing destination with the ODS LISTING CLOSE statement, you will continue to create Listing output.

## Creating Printer Output

If you open the Printer destination, you can create output that is formatted for a high-resolution printer. The first ODS PRINTER statement in the following SAS program opens the Printer destination and directs the formatted output to the file **odsprinter.ps**. The second ODS PRINTER statement closes the Printer destination. You must close the Printer destination before you can print the file.

The data set STATEPOP is created in a DATA step on page 1418. The REGFMT. format is created in a PROC FORMAT step on page 1200. The printer output appears in Display 2.1 on page 20.

```
options nodate nonumber;
ods printer file='odsprinter.ps';
proc tabulate data=statepop;
  class region state;
  var citypop_80 citypop_90;
  table region*state, citypop_80*sum=' ' citypop_90*sum=' ';
  format region regfmt.;
  where region=1;
  label citypop_80='1980' citypop_90='1990';
  title 'Metropolitan Population for the Northeast Region';
  title2 '(measured in millions)';
run;
ods printer close;
```

**Display 2.1**    Output Created by the Printer Destination

### *Metropolitan Population for the Northeast Region (measured in millions)*

		1980	1990
Geographic region	State		
Northeast	CT	2.98	3.15
	MA	5.53	5.79
	ME	0.41	0.44
	NH	0.54	0.66
	NJ	7.37	7.73
	NY	16.14	16.52
	PA	10.07	10.08
	RI	0.89	0.94
	VT	0.13	0.15

## Creating HTML Output

If you open the HTML destination, you can create output that is formatted in HyperText 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 first ODS HTML 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. The second ODS HTML statement closes the HTML destination. You must close the HTML destination before you can browse the HTML files.

```
/* Create HTML files. */
ods html file='odshtml-body.htm'
          contents='odshtml-contents.htm'
          page='odshtml-page.htm'
          frame='ods-html-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 22.

Display 2.2 First View of the Frame File

Table of Contents			
<ul style="list-style-type: none"> <li>■ The Univariate Procedure               <ul style="list-style-type: none"> <li>• CityPop_90                   <ul style="list-style-type: none"> <li>• Moments</li> <li>• Basic Measures of Location and Variability</li> <li>• Tests For Location</li> <li>• Quantiles</li> <li>• Extreme Observations</li> </ul> </li> <li>• NonCityPop_90                   <ul style="list-style-type: none"> <li>• Moments</li> <li>• Basic Measures of Location and Variability</li> <li>• Tests For Location</li> <li>• Quantiles</li> <li>• Extreme Observations</li> <li>• Missing Values</li> </ul> </li> </ul> </li> </ul>			
Table of Pages			
<ul style="list-style-type: none"> <li>■ The Univariate Procedure               <ul style="list-style-type: none"> <li>• Page 1</li> <li>• Page 2</li> </ul> </li> </ul>			

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

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 23.)





```
proc univariate data=statepop mu0=3.5;  
  var citypop_90 noncitypop_90;  
  title;  
run;  
  
ods trace off;
```

**Output 2.1** 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.

```

36  options nodate pageno=1 linesize=64 pagesize=60;
37  ods trace on;
38
39  proc univariate data=statepop mu0=3.5;
40      var citypop_90 noncitypop_90;
41      title;
42  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.ExtremeObs
-----

```

```

Output Added:
-----
Name:      Moments
Label:      Moments
Template:   base.univariate.Moments
Path:      Univariate.NonCityPop_90.Moments
-----

Output Added:
-----
Name:      BasicMeasures    <---
Label:      Basic Measures of Location and Variability
Template:   base.univariate.Measures
Path:      Univariate.NonCityPop_90.BasicMeasures
-----

Output Added:
-----
Name:      TestsForLocation
Label:      Tests For Location
Template:   base.univariate.Location
Path:      Univariate.NonCityPop_90.TestsForLocation
-----

Output Added:
-----
Name:      Quantiles
Label:      Quantiles
Template:   base.univariate.Quantiles
Path:      Univariate.NonCityPop_90.Quantiles
-----

Output Added:
-----
Name:      ExtremeObs
Label:      Extreme Observations
Template:   base.univariate.ExtObs
Path:      Univariate.NonCityPop_90.ExtremeObs
-----

Output Added:
-----
Name:      MissingValues
Label:      Missing Values
Template:   base.univariate.Missings
Path:      Univariate.NonCityPop_90.MissingValues
-----

```

If you compare this SAS log to the Results Folder that appears in Display 2.6 on page 31, 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 “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, the Printer destination, and the Output destination. For more information about ODS destinations, see “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 2.1 on page 24). 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 “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 file='select-body.htm'
         contents='select-contents.htm'
         page='select-page.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.

<b>Table of Contents</b> <ul style="list-style-type: none"> <li>■ The Univariate Procedure <ul style="list-style-type: none"> <li>• CityPop_90 <ul style="list-style-type: none"> <li>• Basic Measures of Location and Variability</li> <li>• Quantiles</li> </ul> </li> <li>• NonCityPop_90 <ul style="list-style-type: none"> <li>• Basic Measures of Location and Variability</li> <li>• Quantiles</li> </ul> </li> </ul> </li> </ul>	
<b>Table of Pages</b> <ul style="list-style-type: none"> <li>■ The Univariate Procedure <ul style="list-style-type: none"> <li>• Page 1</li> <li>• Page 2</li> </ul> </li> </ul>	

The UNIVARIATE Procedure			
Variable: CityPop_90 (1990 metropolitan pop in millions)			
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
Quantiles (Definition 5)			
Quantile	Estimate		
100% Max	28.799		
99%	28.799		
95%	14.166		
90%	9.574		
75% Q3	4.376		
50% Med	2.423		
25% Q1	0.776		
10% Min	0.057		

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

## Creating an Output Data Set

The Output Delivery System 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.

```

/* 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 named MEASURES */
/* 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.

Table of Contents

- The Print Procedure
- Data Set WORK.MEASURES

Output Data Set Produced from  
PROC UNIVARIATE Basic Measures

VarName	LocMeasure	LocValue	VarMeasure	VarValue
CityPop_90	Mean	3.877020	Std Deviation	5.16465
CityPop_90	Median	2.423000	Variance	26.67364
CityPop_90	Mode	.	Range	28.66500
CityPop_90		–	Interquartile Range	3.60000
NonCityPop_90	Mean	1.040429	Std Deviation	0.66036
NonCityPop_90	Median	0.961000	Variance	0.43608
NonCityPop_90	Mode	0.608000	Range	2.75600
NonCityPop_90		–	Interquartile Range	1.12700

For more information about creating output data sets, see the discussion of the ODS OUTPUT statement in “The ODS Statements,” in *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. It marks the link with an icon that identifies the output destination that created the output.

Consider the following SAS program, which generates Listing, HTML, and Printer output as well as an output data set (Output 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 1418 creates this data set.

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

```
ods html file='results-body.htm';
ods printer file='results.ps';
ods output basicmeasures=measures;
```

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

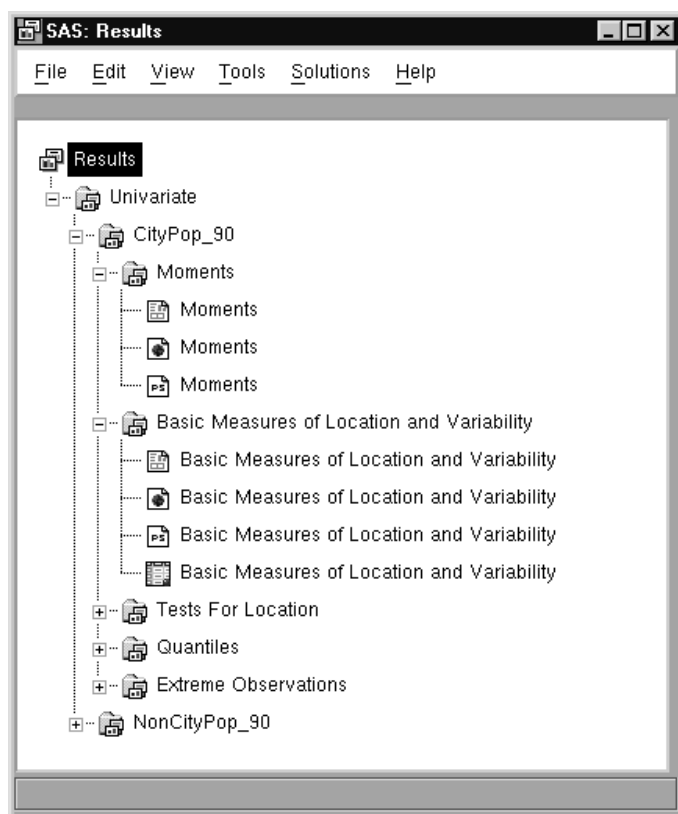
```
ods html close;
ods printer close;
```



The Results folder (see Display 2.6 on page 31) shows the folders and output objects that the procedure produces.

**Display 2.6** View of the Results Folder

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 each contain a folder for each output object. Within the folder for each output object is a link to each piece of output. The icon next to the link indicates which ODS destination created the output. You can see that the Moments object was sent to the Listing, HTML, and Printer destinations while the Basic Measures of Location and Variability was sent to the Listing, HTML, Printer, and Output destinations.



## 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. This section illustrates how to make an alternative table definition. The explanation here focuses on the structure of the table. For detailed explanations of all the statements and attributes that the program uses, see the section on the TEMPLATE procedure in *The Complete Guide to the SAS Output Delivery System*.

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 2.1 on page 24.) 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.

The customized HTML output object appears in Display 2.7 on page 35. The customized Listing output appears in Output 2.2 on page 35. The customized Printer output appears in Display 2.8 on page 36.

```

/* These four options all affect the Listing output. */
/* NODATE and NONUMBER also affect the Printer output.*/
/* None of them affects the HTML output.           */
options nodate nonumber 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;
        style=rowheader;
    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;
        style=rowheader;
    end;

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

    /* End the table definition. */
    end;
    /* Run the procedure. */
    run;

```

```

/* 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 PRINTER statement opens the      */
/* Printer destination and identifies the    */
/* file to write to.                        */
ods printer file='statepop.ps';

/* 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;
/* Close the Printer destination. */
ods printer close;

```

**Display 2.7** Customized HTML Output from PROC UNIVARIATE (partial output)

Table of Contents

1. The Univariate Procedure

CityPop\_90

Basic Measures of Location and Variability

NonCityPop\_90

Basic Measures of Location and Variability

Table of Pages

1. The Univariate Procedure

Page 1

Page 2

The UNIVARIATE Procedure

Variable: CityPop\_90 (1990 metropolitan pop in millions)

Basic Statistical Measures			
Measures of Variability		Measures of Location	
Std Deviation	5.165	Mean	3.877
Variance	26.674	Median	2.423
Range	28.665	Mode	.
Interquartile Range	3.600		

The UNIVARIATE Procedure

Variable: NonCityPop\_90 (1990 nonmetropolitan pop in million)

Basic Statistical Measures			
Measures of Variability		Measures of Location	
Std Deviation	0.660	Mean	1.040
Variance	0.436	Median	0.961
Range	2.756	Mode	0.608
Interquartile Range	1.127		

**Output 2.2** Customized Listing Output from PROC UNIVARIATE

The UNIVARIATE Procedure			
Variable: CityPop_90 (1990 metropolitan pop in millions)			
Basic Statistical Measures			
Measures of Variability		Measures of Location	
Std Deviation	5.165	Mean	3.877
Variance	26.674	Median	2.423
Range	28.665	Mode	.
Interquartile Range	3.600		

The UNIVARIATE Procedure			
Variable: NonCityPop_90 (1990 nonmetropolitan pop in million)			
Basic Statistical Measures			
Measures of Variability		Measures of Location	
Std Deviation	0.660	Mean	1.040
Variance	0.436	Median	0.961
Range	2.756	Mode	0.608
Interquartile Range	1.127		

**Display 2.8** Customized Printer Output from PROC UNIVARIATE (page 1)

***The UNIVARIATE Procedure***  
***Variable: CityPop\_90 (1990 metropolitan pop in millions)***

Basic Statistical Measures			
Measures of Variability		Measures of Location	
Std Deviation	5.165	Mean	3.877
Variance	26.674	Median	2.423
Range	28.665	Mode	.
Interquartile Range	3.600		

**Display 2.9** Customized Printer Output from PROC UNIVARIATE (page 2)

***The UNIVARIATE Procedure***  
***Variable: NonCityPop\_90 (1990 nonmetropolitan pop in million)***

Basic Statistical Measures			
Measures of Variability		Measures of Location	
Std Deviation	0.660	Mean	1.040
Variance	0.436	Median	0.961
Range	2.756	Mode	0.608
Interquartile Range	1.127		

---

## A Gallery of HTML and Printer Output Produced by Base Procedures

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

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

If Printer output is shown, the specified example was run with this ODS PRINTER statement preceding it:

```
ods printer file='external-file';
```

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

```
ods html close;
```

You must execute the following statement before you can print Printer output:

```
ods printer close;
```

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

The SAS program that produces this output is in Example 6 on page 1212.

**Display 2.10**    HTML Output from PROC TABULATE

<i>Energy Expenditures for Each Region (millions of dollars)</i>				
		Customer Base		All Customers
		Residential Customers	Business Customers	
Region	Division			
Northeast	New England	7,477	5,129	12,606
	Middle Atlantic	19,379	15,078	34,457
	Subtotal	26,856	20,207	47,063
West	Division			
	Mountain	5,476	4,729	10,205
	Pacific	13,959	12,619	26,578
	Subtotal	19,435	17,348	36,783
Total for All Regions		\$46,291	\$37,555	\$83,846

**Display 2.11**    Printer Output from PROC TABULATE

<i>Energy Expenditures for Each Region (millions of dollars)</i>					1
		Customer Base		All Customers	
		Residential Customers	Business Customers		
Region	Division				
Northeast	New England	7,477	5,129	12,606	
	Middle Atlantic	19,379	15,078	34,457	
	Subtotal	26,856	20,207	47,063	
West	Division				
	Mountain	5,476	4,729	10,205	
	Pacific	13,959	12,619	26,578	
	Subtotal	19,435	17,348	36,783	
Total for All Regions		\$46,291	\$37,555	\$83,846	

## PROC FREQ: Analyzing a 2×2 Contingency Table

The SAS program that produces this output is in Example 4 on page 580.

Display 2.12 HTML Output from PROC FREQ

**Case-Control Study of High Fat/Cholesterol Diet**

*The FREQ Procedure*

Frequency Percent Row Pct Col Pct	Table of Exposure by Response			
	Exposure	Response(Heart Disease)		Total
		Yes	No	
	High Cholesterol Diet	11 47.83 73.33 84.62	4 17.39 26.67 40.00	15 65.22
	Low Cholesterol Diet	2 8.70 25.00 15.38	6 26.09 75.00 60.00	8 34.78
	Total	13 56.52	10 43.48	23 100.00

*Statistics for Table of Exposure by Response*

Statistic	DF	Value	Prob (Asymptotic)	Prob (Exact)
Chi-Square	1	4.9597	0.0259	0.0393
Likelihood Ratio Chi-Square	1	5.0975	0.0240	
Continuity Adj. Chi-Square	1	3.1879	0.0742	
Mantel-Haenszel Chi-Square	1	4.7441	0.0294	
Fisher's Exact Test (Left)				0.9967
(Right)				0.0367
(2-Tail)				0.0393
Phi Coefficient		0.4644		
Contingency Coefficient		0.4212		
Cramer's V		0.4644		



**Display 2.13**    Printer Output from PROC FREQ (page 1)**Case-Control Study of High Fat/Cholesterol Diet**

1

Frequency Percent Row Pct Col Pct	<i>The FREQ Procedure</i>			
	Table of Exposure by Response			
	Exposure	Response(Heart Disease)		Total
		Yes	No	
	High Cholesterol Diet	11 47.83 73.33 84.62	4 17.39 26.67 40.00	15 65.22
	Low Cholesterol Diet	2 8.70 25.00 15.38	6 26.09 75.00 60.00	8 34.78
	Total	13 56.52	10 43.48	23 100.00

**Statistics for Table of Exposure by Response**

Statistic	DF	Value	Prob
Chi-Square	1	4.9597	0.0259
Likelihood Ratio Chi-Square	1	5.0975	0.0240
Continuity Adj. Chi-Square	1	3.1879	0.0742
Mantel-Haenszel Chi-Square	1	4.7441	0.0294
Phi Coefficient		0.4644	
Contingency Coefficient		0.4212	
Cramer's V		0.4644	

**WARNING: 50% of the cells have expected counts less than 5.  
(Asymptotic) Chi-Square may not be a valid test.**

Pearson Chi-Square Test	
Chi-Square	4.9597
DF	1
Asymptotic Pr > ChiSq	0.0259
Exact Pr >= ChiSq	0.0393

**Display 2.14** Printer Output from PROC FREQ (page 2)**Case-Control Study of High Fat/Cholesterol Diet****2****The FREQ Procedure**

Fisher's Exact Test	
Cell (1,1) Frequency (F)	11
Left-sided Pr $\leq$ F	0.9967
Right-sided Pr $\geq$ F	0.0367
Table Probability (P)	0.0334
Two-sided Pr $\leq$ P	0.0393

Estimates of the Relative Risk (Row1/Row2)			
Type of Study	Value	95% Confidence Limits	
Case-Control (Odds Ratio)	8.2500	1.1535	59.0029
Cohort (Col1 Risk)	2.9333	0.8502	10.1204
Cohort (Col2 Risk)	0.3556	0.1403	0.9009

Odds Ratio (Case-Control Study)	
Odds Ratio	8.2500
Asymptotic Conf Limits	
95% Lower Conf Limit	1.1535
95% Upper Conf Limit	59.0029
Exact Conf Limits	
95% Lower Conf Limit	0.8677
95% Upper Conf Limit	105.5488

**Sample Size = 23****PROC PRINT: Summing Numeric Variables with One BY Group**

The SAS program that produces this output is in Example 4 on page 803.

**Display 2.15**    HTML Output from PROC PRINT

<i>Revenue and Expense Totals for the Northern Region</i>			
State	Month	Expenses	Revenues
NY	FEB95	3,000	4,000
NY	MAR95	6,000	5,000
MA	MAR95	1,500	1,000
Region		10,500	10,000
Number of observations for the state: 3			

<i>Revenue and Expense Totals for the Southern Region</i>			
State	Month	Expenses	Revenues
GA	JAN95	2,000	8,000
GA	FEB95	1,200	6,000
FL	FEB95	8,500	11,000
FL	MAR95	9,800	13,500
Region		21,500	38,500
		32,000	48,500

## **PROC REPORT: Specifying Style Elements for HTML Output in the PROC REPORT Statement**

The SAS program that produces this output is in Example 15 on page 996.

**Display 2.16** HTML Output from PROC REPORT

**Sales for the Southeast Sector**

<i>Manager</i>	<i>Department</i>	<i>Sales</i>
Jones	Paper	40
	Canned	220
	Meat/Dairy	300
	Produce	70
Jones		630
<b>Subtotal for Jones is \$630.00.</b>		
Smith	Paper	50
	Canned	120
	Meat/Dairy	100
	Produce	80
Smith		350
<b>Subtotal for Smith is \$350.00.</b>		
<b>Total for all departments is: \$980.00.</b>		

---

## Customizing the Style Definition That ODS Uses

### What Is a Style Definition?

A style definition determines the overall look of the document that uses it. Each style definition 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 43.


## What Style Definitions Are Shipped with the Software?

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

- 1 Select



- 2 In the Results window, select the Results folder. With your cursor on this folder, use your right mouse button to open the Templates window.
- 3 In the Templates window, select and open **Sashelp.tmplmst**.
- 4 Select **Styles**, and use your right mouse button to open this folder, which contains a list of available style definitions. If you want to view the underlying SAS code for a style definition, select it and open it.

*Operating Environment Information:* For information on navigating in the Explorer window without a mouse, see the section on “Window Controls and General Navigation” in the SAS documentation for your operating environment. 

You can also, submit this PROC TEMPLATE step to see the SAS code for a style definition:

```
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**).

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

In most cases, if you want to alter the style of a 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 table definitions to produce output objects. These table definitions include definitions for table elements: columns, headers, and footers. Each table element 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 and Printer 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 can format their output for HTML or for a high-resolution printer. Their output is in tabular form. Some of the style attributes apply to the table as a whole; others apply to individual cells in the 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 definition that is in use. For information on viewing the attributes in a style, see “What Style Definitions Are Shipped with the Software?” on page 43. The implementation of an attribute depends on the ODS destination that formats the output. In addition, if you are creating HTML output, the implementation of an attribute depends on the browser that you use.  $\triangle$

Many values for style attributes are one of the following:

*'string'*

is a quoted character string.

*dimension*

is a nonnegative number, followed by one of the following units of measure

cm	centimeters
in	inches
mm	millimeters
pt	a printer's point
px	pixels (based on the size of a pixel on the target device)

*Note:* In Version 8 of the SAS System, only the Printer destination supports units of measure on dimensions. However, if you specify CSS in the ODS HTML statement, the HTML destination supports units of measure. The CSS option is experimental in Version 8.  $\triangle$

**Default:** For the HTML destination, pixels; for the Printer destination, units of 1/150 of an inch

*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 predefined SAS color (for example, blue or VIYG)
  - ☐ 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 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.  $\triangle$

- 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:

Lightness	Saturation	Hue
black	gray	blue
very dark	grayish	purple
dark	moderate	red
medium	strong	orange   brown
light	vivid	yellow
very light		green
white		

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.  $\triangle$

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*.

*format*

is a SAS format or a user-defined format.

*reference*

is a reference to an attribute that is defined in the current style or in the parent (or beyond). In this case, the value that you use is the name of the style element followed, in parentheses, by the name of an attribute name within that element. For example, suppose that you create a style element called **DATACELL** that uses the **FOREGROUND=** and **BACKGROUND=** style elements this way:

```
style datacell / background=blue
                  foreground=white;
```

Later, you can ensure that another style element, **NEWCELL**, uses the same background color by defining it this way:

```
style newcell / background=datacell(background);
```

Similarly, suppose that you create a style element called **HIGHLIGHTING** that defines three attributes this way:

```
style highlighting /
  "go"=green
  "caution"=yellow
  "stop"=red;
```

Later, you can define a style element called **MESSAGES** that uses the colors that are defined in **HIGHLIGHTING**:

```
style messages;
  "note"=highlighting("go")
  "warning"=highlighting("caution")
  "error"=highlighting("stop");
```

In this way, multiple style elements could use the colors that you define in **HIGHLIGHTING**. If you decide to change the value of **'go'** to blue, you simply change its value in the definition of **HIGHLIGHTING**, and every style element that references **highlighting** ("go") will use blue instead of green.

*Note:* In the first example, the style attribute **BACKGROUND=** is a predefined style attribute. Therefore, when you reference it, you do not put it in quotation marks. However, in the second example, **'go'** is a user-defined attribute. You define it with quotation marks, and when you reference it, you must use quotation marks. (This section describes all the predefined style attributes that are available.)  $\Delta$

You can use a special form of reference to get a value for a style attribute from the macro table at the time that the style element is used. For instance, the following **STYLE** statement uses the current value of the macro variable **bkgr** for the background color of the style element **cell**:

```
style cell / background=symget("bkgr");
```



*font-definition*

A value can also be a font definition. A font definition has the following general format:

(“*font-face-1* <... , *font-face-n*>”, *font-size*, *keyword-list*)

If you specify only one font face and if its name does not include a space character, you can omit the quotation marks. If you specify more than one font face, the browser or printer uses the first one that is installed on your system.

*font-size* specifies the size of the font. *font-size* can be a dimension or a number without units of measure. If you specify a dimension, you must specify a unit of measure. Without a unit of measure, the number becomes a size that is relative to all other font sizes in the document.

*keyword-list* specifies the weight, font style, and font width. You can include one value for each, in any order. The following table shows the keywords that you can use:

Keywords for Font Weight	Keywords for Font Style	Keywords for Font Width <sup>1</sup>
MEDIUM	ITALIC	NORMAL*
BOLD	ROMAN	COMPRESSED*
DEMI_BOLD*	SLANT	EXTRA_COMPRESSED*
EXTRA_BOLD*		NARROW*
LIGHT		WIDE*
DEMI_LIGHT*		EXPANDED*
EXTRA_LIGHT*		

1    \* Most fonts do not honor these values.

*Note:* You can use the value `_UNDEF_` for any style attribute. ODS treats an attribute that is set to `_UNDEF_` as if its value had never been set, even in the parent or beyond.  $\triangle$

In the list of style attributes that follows, any attribute that is not documented as applying to a particular destination applies to all destinations that support the `STYLE=` option in the ODS statement that opens the destination. In Version 8 of the SAS System, the two destinations that support `STYLE=` are the HTML destination and the Printer destination.

**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.

**Tip:** Generally, the background color of the cell overrides the background color of the table. You see the background color for the table only as the space between cells (see `CELLSPACING=` on page 48).

**Applies to:** 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:** tables or cells

**ODS Destinations:** HTML

**BORDERCOLOR=***color*

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

**Applies to:** 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:** tables or cells

**ODS Destinations:** HTML

**BORDERCOLORLIGHT=***color*

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

**Applies to:** tables or cells

**ODS Destinations:** HTML

**BORDERWIDTH=***dimension*

specifies the width of the border of the table.

**Applies to:** tables

**Tip:** Typically, when **BORDERWIDTH=0**, the ODS destination sets **RULES=NONE** (see the discussion of **RULES=** on page 53) and **FRAME=VOID** (see the discussion of **FRAME=** on page 50).

**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

**ODS Destinations:** HTML

**CELLPADDING=***dimension | integer%*

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

**Applies to:** tables

**CELLSPACING=***dimension*

specifies the thickness of the spacing between cells.

**Applies to:** tables

**Interaction:** If **BORDERWIDTH=** is nonzero, and if the background color of the cells contrasts with the background color of the table, the color of the cell spacing is determined by 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:** The ODS destination 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

**ODS Destinations:** HTML

FONT=*font-definition*

specifies a font definition to use. For more information, see the discussion of font definition on page 47.

**Applies to:** cells

FONT\_FACE=*'string-1<... , string-n>'*

specifies the font face to use. If you supply more than one string, the browser or printer uses the first one that is installed on your system.

You cannot be sure what fonts are available to someone who is viewing your output in a browser or printing it on a high-resolution printer. Most devices support

- ☐ times
- ☐ courier
- ☐ arial, helvetica.

**Applies to:** cells

FONT\_SIZE=*dimension* | *size*

specifies the size of the font. The value of *size* is relative to all other font sizes in the document.

**Applies to:** cells

**Range:** 1 to 7, for *size*

**Restriction:** If you specify a dimension, you must specify a unit of measure.

Without a unit of measure, the number becomes a relative size.

FONT\_STYLE=ITALIC | ROMAN | SLANT

specifies the style of the font. In many cases, italic and slant map to the same font.

**Applies to:** 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

**Applies to:** cells

**Restriction:** You cannot be sure what font weights are available to someone who is viewing your output in a browser or printing it on a high-resolution printer. 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:** cells

**Restriction:** Most fonts do not honor these values.

FOREGROUND=*color*

specifies the color of the foreground, which is primarily the color of text.

**Applies to:** tables or cells

FRAME=*frame-type*

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

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

**Applies to:** tables

HREFTARGET=*target*

specifies the window or frame in which to open the target of the link. *target* can be

\_BLANK

opens the target in a new, blank window. The window has no name.

\_PARENT

opens the target in the window from which the current window was opened.

\_SEARCH

opens the target in the browser's search pane.

Restriction: Available only in Internet Explorer 5 or later.

\_SELF

opens the target in the current window.

\_TOP

opens the target in the topmost window.

'*name*'

opens the target in the specified window or the frame.

**Default:** `_SELF`

**Applies to:** cells

**ODS Destinations:** HTML

`HTMLCLASS=string`

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

**Applies to:** tables and cells

**ODS Destinations:** HTML

`HTMLID=string`

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

**Applies to:** tables and cells

**ODS Destinations:** HTML

`HTMLSTYLE=string`

specifies individual attributes and values for the table or cell.

**Applies to:** tables and cells

**ODS Destinations:** HTML

`JUST=justification`

specifies justification, where *justification* can be

**CENTER**

specifies center justification.

Alias: C

Applies to: tables and cells

**LEFT**

specifies left justification.

Alias: L

Applies to: tables and cells

**RIGHT**

specifies right justification.

Alias: R

Applies to: tables and 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 allow SAS to break a line at a space character.

**OFF**

allows SAS to break a line at a space character if appropriate.

**Applies to:** cells

`OUTPUTWIDTH=dimension | integer%`

specifies the width of the table. If you specify a percent, it represents a percentage of the width of the browser window.

**Applies to:** tables

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

**ODS Destinations:** HTML

POSTHTML=*'string'*

specifies the HTML code to place after the table or cell.

**Applies to:** tables or cells

**ODS Destinations:** HTML

POSTIMAGE=*'string'*

specifies an image to place after the 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:** tables or cells

**ODS Destinations:** HTML

POSTTEXT=*'string'*

specifies text to place after the cell or table.

**Applies to:** tables or cells

PREHTML=*'string'*

specifies the HTML code to place before the table or cell.

**Applies to:** tables or cells

**ODS Destinations:** HTML

PREIMAGE=*'string'*

specifies an image to place before the 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:** tables or cells

**ODS Destinations:** HTML

PRETEXT=*'string'*

specifies text to place before the cell or table.

**Applies to:** 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 precede the <, spaces that immediately follow 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:** tables or cells

**ODS Destinations:** HTML

**RULES=***rule-type*

specifies the types of rules to use in a table. The following table shows the possible values of *rule* and their meanings.

This value of <i>rule</i>	Creates rules in these locations
ALL	between all rows and columns
COLS	between all columns
GROUP	between the table header and the table and between the table and the table footer, if there is one
NONE	no rules anywhere
ROWS	between all rows

**Applies to:** tables

**TAGATTR=***'string'*

specifies text to insert in the HTML. The string must be valid HTML for the context in which the style element is rendered. Many style elements are rendered between `<TD>` and `</TD>` tags. To determine how a style element is rendered, look at the source for the output.

**Applies to:** cells

**ODS Destinations:** HTML

**URL=***'uniform-resource-locator'*

specifies a URL to link to from the current cell.

**Applies to:** cells

**ODS Destinations:** HTML

**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.

Alias: M

**Applies to:** cells

---

## 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:

CATALOG  
 DATASETS  
 PLOT  
 PMENU  
 TRANTAB

See the section on the individual procedure for more information.

*Note:* PROC SQL executes each query automatically. Neither the RUN nor RUN CANCEL statement has any effect. △

---

## 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  $n$ th BY variable in the BY statement.

*BY-variable*

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



**BYVAR $n$  | 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  $n$ th 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 1504.
- 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; ❶
    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
;

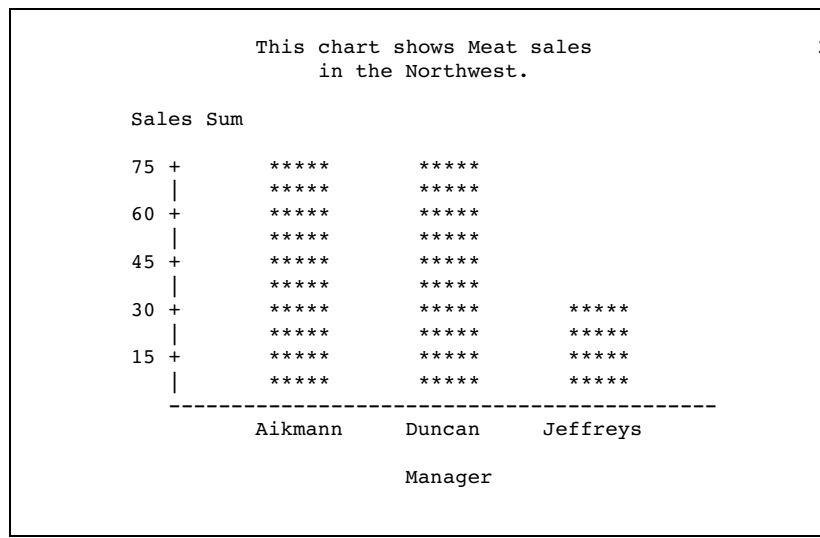
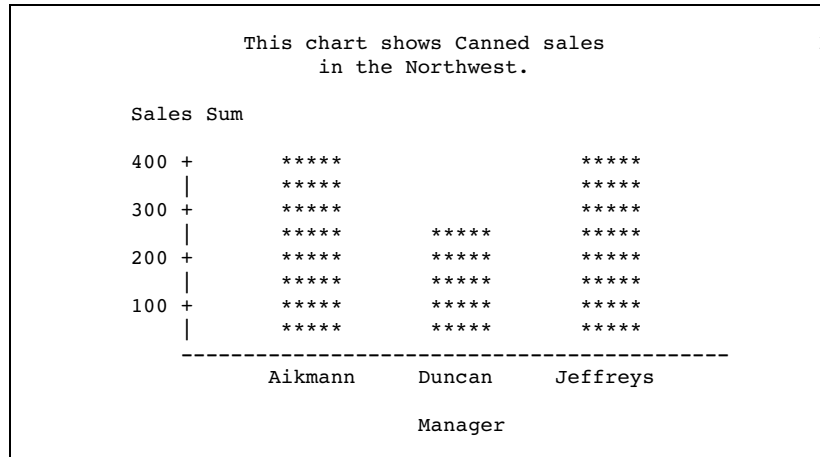
proc sort data=groc; ❷
    by region department;
run;
options nobyline nodate pageno=1
        linesize=64 pagesize=20; ❸
```

```

proc chart data=groc; ❹
  by region department;
  vbar manager / type=sum sumvar=sales;
  title1 'This chart shows #byval2 sales';
  title2 'in the #byval(region)..';
run;
options byline; ❺

```

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



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

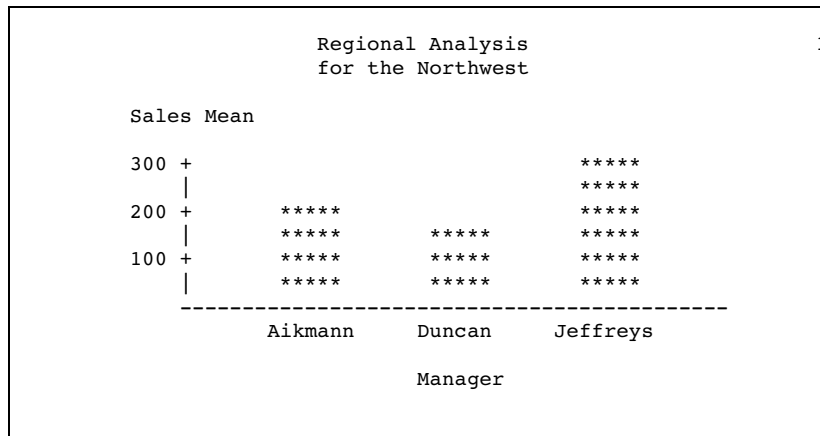
This example inserts the name of a BY variable and the value of a BY variable into the title. The program

- 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 a1 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; ❶
proc chart data=groc; ❷
  by region;
  vbar manager / type=mean sumvar=sales;
  title1 '#byvar(region).al Analysis';
  title2 'for the #byvall';
run;
options byline; ❸
```

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



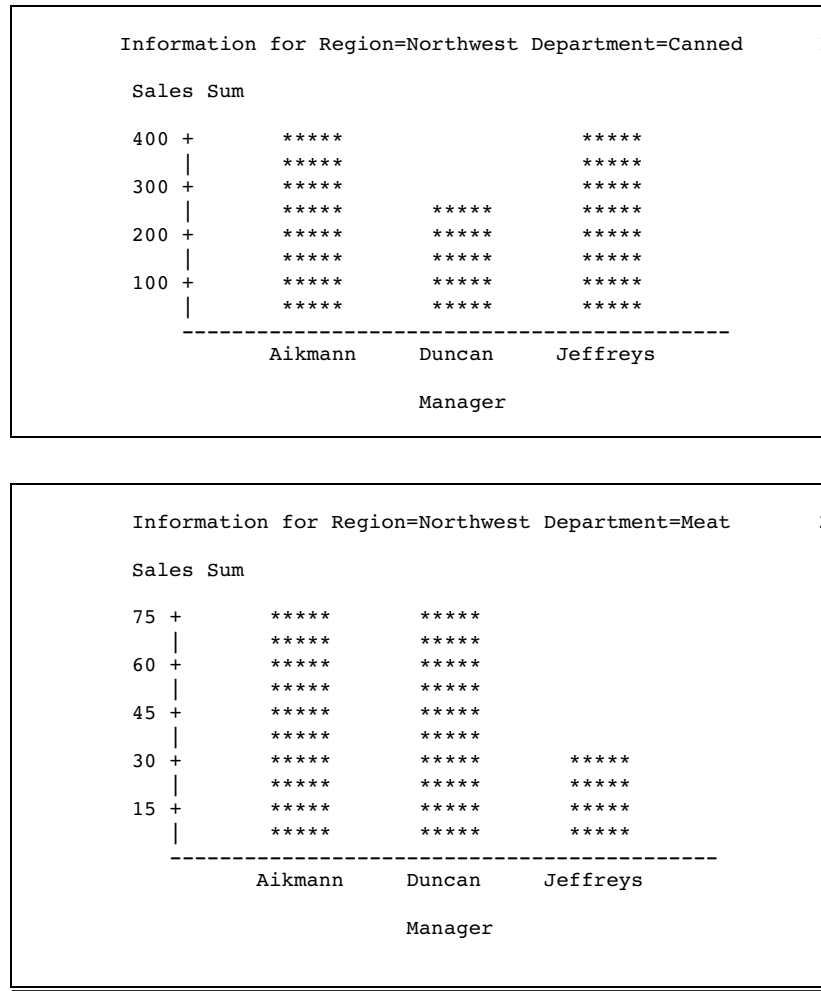
### Example: Inserting the Complete BY Line into a Title

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; ❶
proc chart data=groc; ❷
  by region department;
  vbar manager / type=sum sumvar=sales;
  title 'Information for #byline';
run;
options byline; ❸
```

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



## 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
<b>x1-xn</b>	specifies variables X1 through Xn. The numbers must be consecutive.
<b>x:</b>	specifies all variables that begin with the letter X.
<b>x--a</b>	specifies all variables between X and A, inclusive. This notation uses the position of the variables in the data set.
<b>x-numeric-a</b>	specifies all numeric variables between X and A, inclusive. This notation uses the position of the variables in the data set.

Notation	Meaning
<b>x-character-a</b>	specifies all character variables between X and A, inclusive. This notation uses the position of the variables in the data set.
<b>_numeric_</b>	specifies all numeric variables.
<b>_character_</b>	specifies all character variables.
<b>_all_</b>	specifies all variables.

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

See *SAS Language Reference: Concepts* for complete documentation.

---

## 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 1512 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.

```
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:

PROCLIB.PAYROLL						1
First 10 Observations Only						
Id	Sex	Jobcode	Salary	Birth	Hired	
Number						
1919	M	TA2	34376	12SEP60	04JUN87	
1653	F	ME2	35108	15OCT64	09AUG90	
1400	M	ME1	29769	05NOV67	16OCT90	
1350	F	FA3	32886	31AUG65	29JUL90	
1401	M	TA3	38822	13DEC50	17NOV85	
1499	M	ME3	43025	26APR54	07JUN80	
1101	M	SCP	18723	06JUN62	01OCT90	
1333	M	PT2	88606	30MAR61	10FEB81	
1402	M	TA2	32615	17JAN63	02DEC90	
1479	F	TA3	38785	22DEC68	05OCT89	

The following PROC FORMAT step creates the format \$JOBfmt., 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 \$JOBfmt. format with the variable Jobcode:

```
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:

Summary Statistics for Each Job Code				1
The MEANS Procedure				
Analysis Variable : Salary				
Jobcode	N Obs	Mean	Maximum	
Baggage Checker	9	25794.22	26896.00	
Flight Attendant Trainee	11	23039.36	23979.00	
Junior Flight Attendant	16	27986.88	28978.00	
Senior Flight Attendant	7	32933.86	33419.00	
Mechanic Trainee	8	28500.25	29769.00	
Junior Mechanic	14	35576.86	36925.00	
Senior Mechanic	7	42410.71	43900.00	
Junior Navigator	5	42032.20	43433.00	
Senior Navigator	3	52383.00	53798.00	
Pilot Trainee	8	67908.00	71349.00	
Junior Pilot	10	87925.20	91908.00	
Senior Pilot	2	10504.50	11379.00	
Skycap	7	18308.86	18833.00	
Ticket Agent Trainee	9	27721.33	28880.00	
Junior Ticket Agent	20	33574.95	34803.00	
Senior Ticket Agent	12	39679.58	40899.00	
-----				

*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:

Summary Statistics for Job Codes (Using a Format that Groups the Job Codes)				1
The MEANS Procedure				
Analysis Variable : Salary				
Jobcode	N Obs	Mean	Maximum	
Baggage Checker	9	25794.22	26896.00	
Flight Attendant	34	27404.71	33419.00	
Mechanic	29	35274.24	43900.00	
Navigator	8	45913.75	53798.00	
Pilot	20	72176.25	91908.00	
Skycap	7	18308.86	18833.00	
Ticket Agent	41	34076.73	40899.00	

### 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:

```

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 with the Variable Salary						1
Id Number	Sex	Jobcode	Salary	Birth	Hired	
1919	M	TA2	\$34,376	12SEP60	04JUN87	
1653	F	ME2	\$35,108	15OCT64	09AUG90	
1400	M	ME1	\$29,769	05NOV67	16OCT90	
1350	F	FA3	\$32,886	31AUG65	29JUL90	
1401	M	TA3	\$38,822	13DEC50	17NOV85	
1499	M	ME3	\$43,025	26APR54	07JUN80	
1101	M	SCP	\$18,723	06JUN62	01OCT90	
1333	M	PT2	\$88,606	30MAR61	10FEB81	
1402	M	TA2	\$32,615	17JAN63	02DEC90	
1479	F	TA3	\$38,785	22DEC68	05OCT89	

### 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:

Average GPA			1
The MEANS Procedure			
Analysis Variable : GPA			
Year	N Obs	Mean	
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 456. △

---

## 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 815 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

1491contains 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 65 identifies common descriptive statistics that are available in several base procedures. See “Keywords and Formulas” on page 1458 for more detailed information about available statistics and theoretical information.

**Table 2.1** Common Descriptive Statistics That Base Procedures Calculate

Statistic	Description	Procedures
confidence intervals		FREQ, MEANS, UNIVARIATE
CSS	corrected sum of squares	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
CV	coefficient of variation	MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
goodness-of-fit tests		FREQ, UNIVARIATE
KURTOSIS	kurtosis	MEANS/SUMMARY, UNIVARIATE
MAX	largest (maximum) value	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
MEAN	mean	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
MEDIAN	median (50 <sup>th</sup> percentile)	CORR (for nonparametric correlation measures), MEANS/SUMMARY, TABULATE, UNIVARIATE
MIN	smallest (minimum) value	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
MODE	most frequent value (if not unique, the smallest mode is used)	UNIVARIATE
N	number of observations on which calculations are based	CORR, FREQ, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
NMISS	number of missing values	FREQ, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
NOBS	number of observations	MEANS/SUMMARY, UNIVARIATE
PCTN	the percentage of a cell or row frequency to a total frequency	REPORT, TABULATE
PCTSUM	the percentage of a cell or row sum to a total sum	REPORT, TABULATE
Pearson correlation		CORR
percentiles		FREQ, MEANS/SUMMARY, TABULATE, UNIVARIATE

Statistic	Description	Procedures
RANGE	range	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
robust statistics	trimmed means, Winsorized means	UNIVARIATE
SKEWNESS	skewness	MEANS/SUMMARY, UNIVARIATE
Spearman correlation		CORR
STD	standard deviation	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
STDERR	the standard error of the mean	MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
SUM	sum	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
SUMWGT	sum of weights	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
tests of location		UNIVARIATE
USS	uncorrected sum of squares	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
VAR	variance	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE

## Computational Requirements for Statistics

The following requirements are computational requirements for the statistics that are listed in Table 2.1 on page 65. They do not describe recommended sample sizes.

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

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., SAS® *Procedures Guide, Version 8*, Cary, NC: SAS Institute Inc., 1999. 1729 pp.

**SAS® Procedures Guide, Version 8**

Copyright © 1999 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-482-9

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

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

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

1st printing, October 1999

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

IBM® and DB2® are registered trademarks or trademarks of International Business Machines Corporation. ORACLE® is a registered trademark of Oracle Corporation. ® indicates USA registration.

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

The Institute is a private company devoted to the support and further development of its software and related services.