Paper 154-2010

Using PROC SGPLOT for Quick High-Quality Graphs

Susan J. Slaughter, Avocet Solutions, Davis, CA
Lora D. Delwiche, University of California, Davis, CA

ABSTRACT

New with SAS® 9.2, ODS Graphics introduces a whole new way of generating high-quality graphs using SAS. With just a few lines of code, you can add sophisticated graphs to the output of existing statistical procedures, or create stand-alone graphs. The SGPLOT procedure produces a variety of graphs including bar charts, scatter plots, and line graphs. This paper shows how to produce several types of graphs using PROC SGPLOT, and how to create paneled graphs by converting PROC SGPLOT to PROC SGPANEL. This paper also shows how to send your graphs to different ODS destinations, how to apply ODS styles to your graphs, and how to specify properties of graphs, such as format, name, height, and width. Last, this paper shows how to use the SAS/GRAPH® ODS Graphics Editor to make one-time changes to graphs..

INTRODUCTION

When ODS Graphics was originally conceived, the goal was to enable statistical procedures to produce sophisticated graphs tailored to each specific statistical analysis, and to integrate those graphs with the destinations and styles of the Output Delivery System. In SAS 9.2, over 60 statistical procedures have the ability to produce graphs using ODS Graphics. A fortuitous side effect of all this graphical power has been the creation of a set of procedures for creating stand-alone graphs (graphs that are not embedded in the output of a statistical procedure). These procedures all have names that start with the letters SG (SGPLOT, SGSCATTER, SGPANEL, and SGRENDER).

This paper focuses on one of those procedures, the SGPLOT procedure. PROC SGPLOT produces many types of graphs. In fact, this one procedure produces 16 different types of graphs. PROC SGPLOT creates one or more graphs and overlays them on a single set of axes. (There are four axes in a set: left, right, top, and bottom.) Other SG procedures create panels with multiple sets of axes, or render graphs using custom ODS graph templates. Because the syntax for SGPLOT and SGPANEL are so similar, we also show an example of SGPANEL which produces a panel of graphs all using the same axis specifications.

This paper was written using SAS 9.2 Phase 2, but almost all the features discussed here also work in SAS 9.2 Phase1. For those features that are new with Phase 2, we note the differences in their descriptions.

ODS GRAPHICS VS. TRADITIONAL SAS/GRAPH

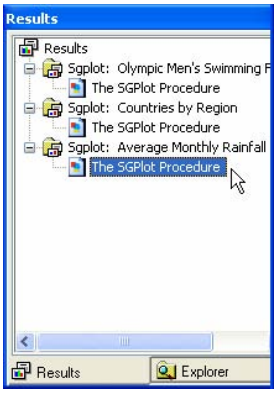
To use ODS Graphics you must have SAS/GRAPH software which is licensed separately from Base SAS. Some people may wonder whether ODS Graphics replaces traditional SAS/Graph procedures. No doubt, for some people and some applications, it will. But ODS Graphics is not designed to do everything that traditional SAS/Graph does, and does not replace it. For example, ODS Graphics does not create contour plots; for contour plots you need to use traditional SAS/GRAPH.

Here are some of the major differences between ODS Graphics and traditional SAS/GRAPH procedures.

Traditional SAS/GRAPH	ODS Graphics
• Graphs are saved in SAS graphics catalogs	• Produces graphs in standard image file formats such as PNG and JPEG
• Graphs are viewed in the Graph window	• Graphs are viewed in standard viewers such as a web browser for HTML output
• Can use GOPTIONS statements to control appearance of graphs	• GOPTIONS statements have no effect

VIEWING ODS GRAPHICS

When you produce ODS Graphics in the SAS windowing environment, for most output destinations the Results Viewer window opens to display your results. However, when you use the LISTING destination, graphs are not automatically displayed. You can always view graphs, regardless of their destination, by double-clicking their graph icons in the Results window.



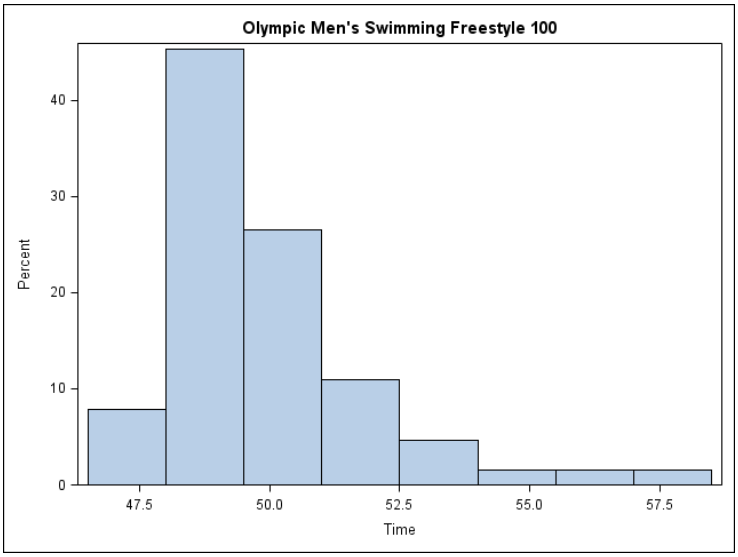
EXAMPLES

The following examples show a small sample of the types of graphs the SGPLOT procedure can produce.

HISTOGRAMS

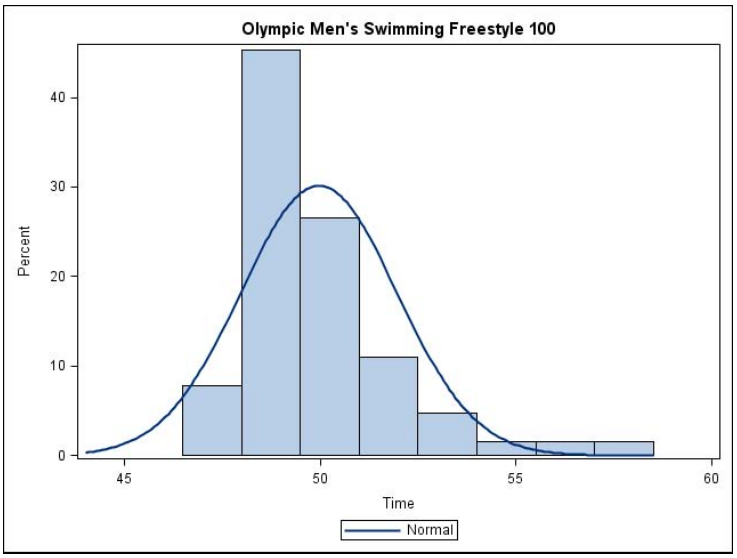
Histograms show the distribution of a continuous variable. The following PROC SGPLOT uses data from the preliminary heats of the 2008 Olympic Men's Swimming Freestyle 100 m event. The histogram shows the distribution of the variable, TIME, which is the time in seconds for each swimmer.

```
* Histograms;
PROC SGPLOT DATA = Freestyle;
   HISTOGRAM Time;
   TITLE "Olympic Men's Swimming Freestyle 100";
RUN;
```



The next PROC SGPLOT uses a DENSITY statement to overlay a density plot on top of the histogram. The default density plot is the normal distribution. When overlaying plots, the order of the statements determines which plot is drawn on top. The plot resulting from the first statement will be on the bottom, followed by the second, and so on. Care must be taken to make sure that the subsequent plots do not obscure the first.

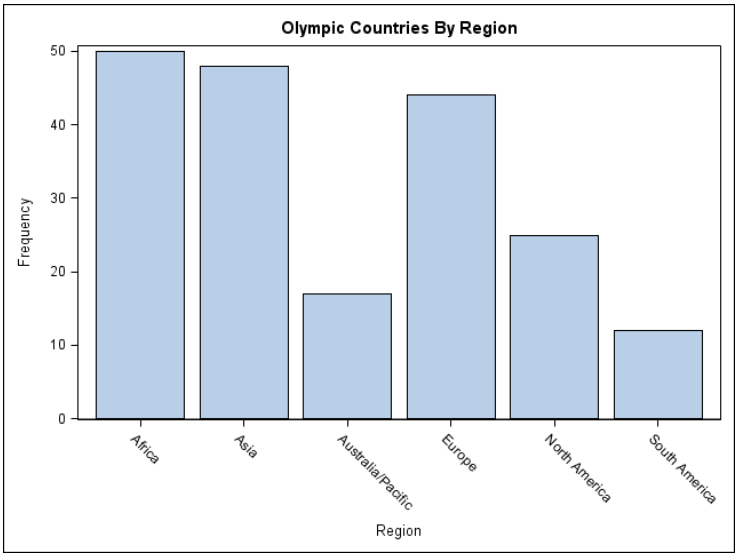
```
PROC SGPLOT DATA = Freestyle;
   HISTOGRAM Time;
   DENSITY Time;
   TITLE "Olympic Men's Swimming Freestyle 100";
RUN;
```



BAR CHARTS

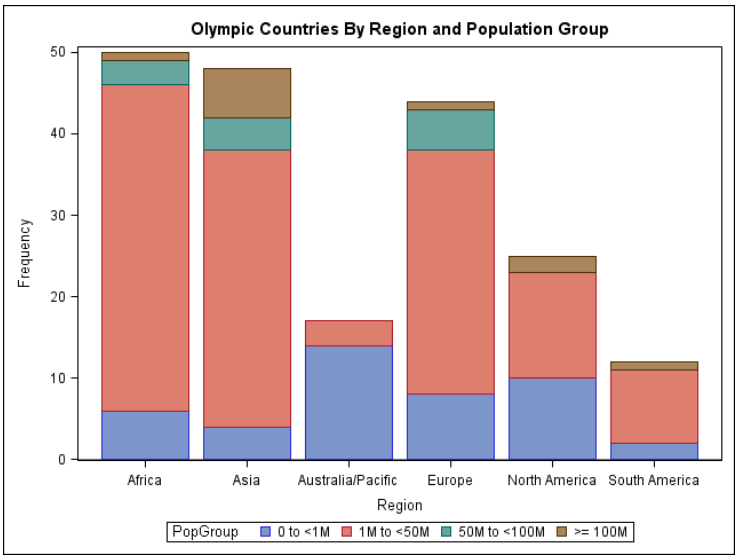
Bar charts show the distribution of a categorical variable. This code uses a VBAR statement to create a vertical bar chart of the variable REGION. The chart shows the number of countries in each region that participated in the 2008 Olympics.

```
* Bar Charts;
PROC SGPLOT DATA = Countries;
   VBAR Region;
   TITLE 'Olympic Countries by Region';
RUN;
```



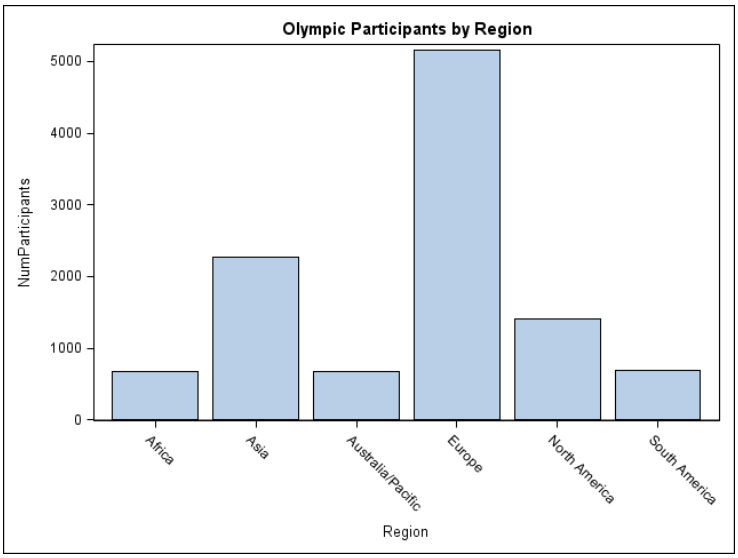
This bar chart is like the first one except that the bars have been divided into groups using the GROUP= option. The grouping variable is a categorical variable named POPGROUP. The GROUP= option can be used with many SGPLOT statements (see Table 1).

```
PROC SGPLOT DATA = Countries;
   VBAR Region / GROUP = PopGroup;
   TITLE 'Olympic Countries by Region and Population Group';
RUN;
```



In the following code, the GROUP= option has been replaced with a RESPONSE= option. The response variable is NUMPARTICIPANTS, the number of participants in the 2008 Olympics from each country. Now each bar represents the total number of participants for a region.

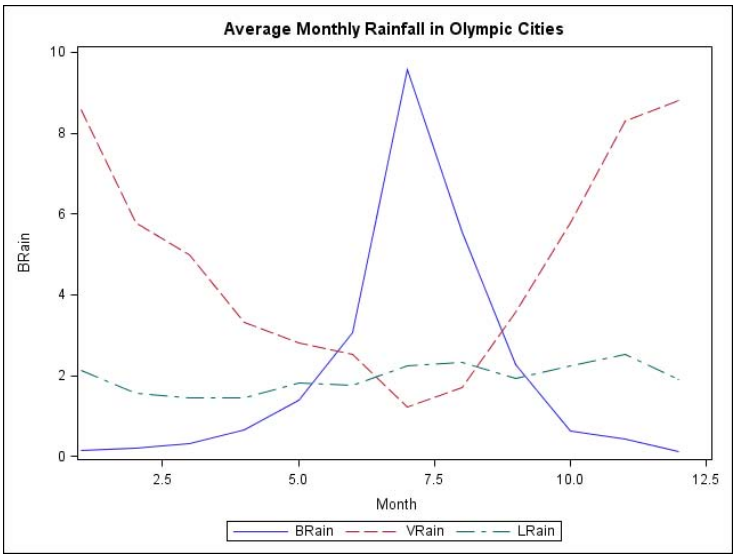
```
PROC SGPLOT DATA = Countries;
   VBAR Region / RESPONSE = NumParticipants;
   TITLE 'Olympic Participants by Region';
RUN;
```



SERIES PLOTS

In a series plot, the data points are connected by a line. This example uses the average monthly rainfall for three cities, Beijing, Vancouver, and London. Three SERIES statements overlay the three lines. Data for series plots must be sorted by the X variable. If your data are not already in the correct order, then use PROC SORT to sort the data before running the SGPLOT procedure.

```
* Series plot;
PROC SGPLOT DATA = Weather;
   SERIES X = Month Y = BRain;
   SERIES X = Month Y = VRain;
   SERIES X = Month Y = LRain;
   TITLE 'Average Monthly Rainfall in Olympic Cities';
RUN;
```



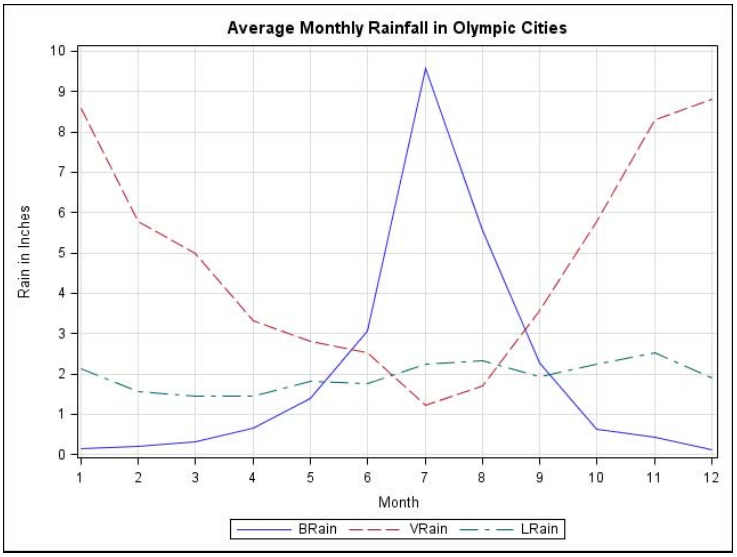
EMBELLISHING GRAPHS

So far the examples have shown how to create basic graphs. The remaining examples show statements and options you can use to change the appearance of your graphs.

XAXIS AND YAXIS STATEMENTS

In the preceding series plot, the variable on the X axis is Month. The values of Month are integers from 1 to 12, but the default labels on the X axis have values like 2.5. In the following code, the option TYPE = DISCRETE tells SAS to use the actual data values. Other options change the axis label and set values for the Y axis, and add grid lines.

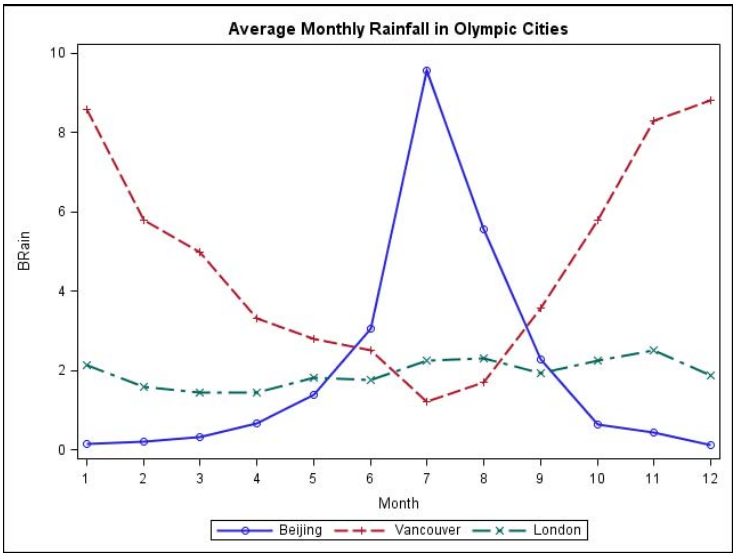
```
* Plot with XAXIS and YAXIS;
PROC SGPLOT DATA = Weather;
   SERIES X = Month Y = BRain;
   SERIES X = Month Y = VRain;
   SERIES X = Month Y = LRain;
   XAXIS TYPE = DISCRETE GRID;
   YAXIS LABEL = 'Rain in Inches' GRID VALUES = (0 TO 10 BY 1);
   TITLE 'Average Monthly Rainfall in Olympic Cities';
RUN;
```



PLOT STATEMENT OPTIONS

Many options can be added to plot statements. For these SERIES statements, the options LEGENDLABEL=, MARKERS, AND LINEATTRS= have been added. The LEGENDLABEL= option can be used with any of the plot statements, while the MARKERS and LINEATTRS= options can only be used with certain plot statements (see Table 1).

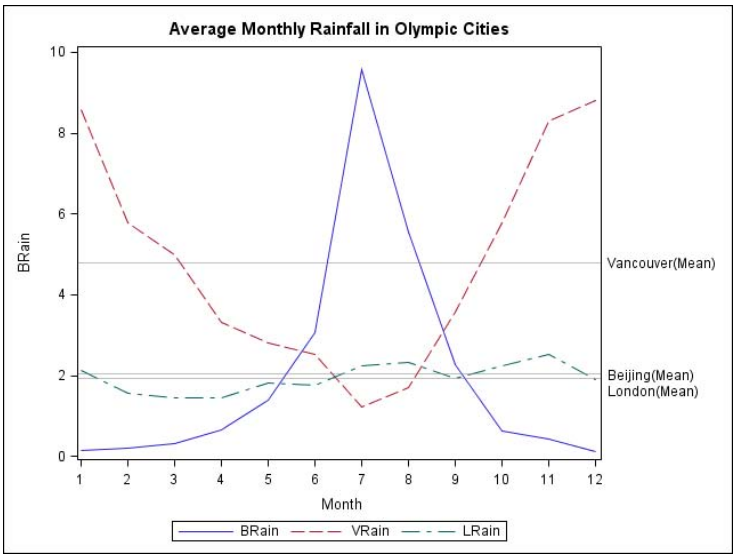
```
* Plot with options on plot statements;
PROC SGPLOT DATA = Weather;
   SERIES X = Month Y = BRain / LEGENDLABEL = 'Beijing'
      MARKERS LINEATTRS = (THICKNESS = 2);
   SERIES X = Month Y = VRain / LEGENDLABEL = 'Vancouver'
      MARKERS LINEATTRS = (THICKNESS = 2);
   SERIES X = Month Y = LRain / LEGENDLABEL = 'London'
      MARKERS LINEATTRS = (THICKNESS = 2);
   XAXIS TYPE = DISCRETE;
   TITLE 'Average Monthly Rainfall in Olympic Cities';
RUN;
```



REFLINE STATEMENT

Reference lines can be added to any type of graph. In this case, lines have been added marking the average rainfall per month for the entire year for each city. The TRANSPARENCY= option on the REFLINE statement specifies that the reference line should be 50% transparent. The TRANSPARENCY option can also be used with many other plot statements (see Table 1).

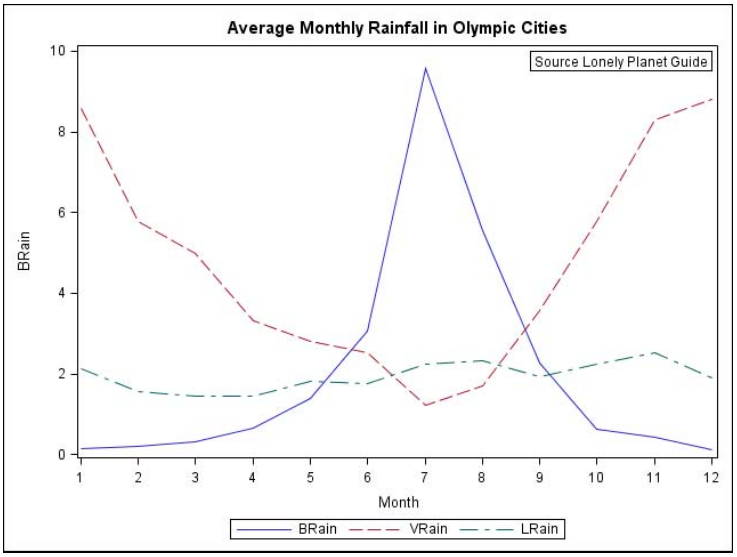
```
* Plot with REFLINE;
PROC SGPLOT DATA = Weather;
   SERIES X = Month Y = BRain;
   SERIES X = Month Y = VRain;
   SERIES X = Month Y = LRain;
   XAXIS TYPE = DISCRETE;
   REFLINE 2.03 4.78 1.94 / TRANSPARENCY = 0.5
     LABEL = ('Beijing(Mean)' 'Vancouver(Mean)' 'London(Mean)');
   TITLE 'Average Monthly Rainfall in Olympic Cities';
RUN;
```



INSET STATEMENT

The INSET statement allows you to add descriptive text to graphs. Insets can be added to any type of graph.

```
* Plot with INSET;
PROC SGPLOT DATA = Weather;
   SERIES X = Month Y = BRain;
   SERIES X = Month Y = VRain;
   SERIES X = Month Y = LRain;
   XAXIS TYPE = DISCRETE;
   INSET 'Source Lonely Planet Guide'/ POSITION = TOPRIGHT BORDER;
   TITLE 'Average Monthly Rainfall in Olympic Cities';
RUN;
```



SGPLOT PROCEDURE SYNTAX

The four tables spread over the next five pages summarize the statements and options for the SGPLOT procedure.

The SGPLOT procedure produces 16 different types of plots that can be grouped into five general areas: basic X Y plots, band plots, fit and confidence plots, distribution graphs for continuous DATA, and distribution graphs for categorical DATA. The VECTOR statement is new with SAS 9.2 Phase 2; all the others are available with SAS 9.2 Phase 1. Many of these plot types can be used together in the same graph. In the preceding examples, we used the HISTOGRAM and DENSITY statements together to produce a histogram overlaid with a normal density curve. We also used three SERIES statements together to produce one graph with three different series lines. However, not all plot statements can be used with all other plot statements. Table 1 shows which statements can be used with which other statements. Table 1 also includes several options that can be used with many different plot statements.

Table 2 shows each of the 16 plot statements along with their basic syntax and selected options. The options listed in Table 2 are in addition to the options listed in Table 1.

In addition to the plot statements, there are some optional statements you might want to use. These statements can be used with any type of plot to control axes, or add reference lines and insets. Table 3 shows these statements with selected options.

Several types of plots can use the LINEATTR, MARKERATTR, or FILLATTR options to change the appearance of lines, markers, or fill (see Table 1). These options allow you choose values for the color of fill; the color, pattern and thickness of lines; and color, symbol, and size of markers. Table 4 gives the syntax for hard coding the values for these options. Note that it is also possible to use ODS styles to control these attributes, or to change them using the ODS Graphics Editor.

Even with all the options listed in these four tables, this is just a sample. Each plot statement has many possible options—we have listed only a few of the most useful. For a complete list of available options, see the SAS Help and Documentation for PROC SGPLOT.

Table 1. Compatibility of SGPLOT procedure statements and selected options. If a check mark appears in the box then the two statements or options can be used together.

	SCATTER	SERIES	STEP	NEEDLE	VECTOR	BAND	REG	LOESS	PBSPLINE	ELLIPSE	HBOX/VBOX	HISTOGRAM	DENSITY	HBAR/VBAR	HLINE/VLINE	DOT
Basic X Y Plots *PLOTNAME* X=*var* Y=*var* / *options*;																
SCATTER	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓						
SERIES	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓						
STEP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓						
NEEDLE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓						
VECTOR	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓			
Band Plots BAND X=var UPPER=var LOWER=var / options; (You can also specify numeric values for upper and lower.)																
BAND	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓						
Fit and Confidence Plots *PLOTNAME* X=*var* Y=*var* / *options*;																
REG	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓						
LOESS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓						
PBSPLINE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓						
ELLIPSE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓						
Distribution Graphs – Continuous DATA *PLOTNAME response-var / options*;																
HBOX or VBOX																
HISTOGRAM					✓								✓			
DENSITY					✓							✓	✓			
Distribution Graphs – Categorical DATA *PLOTNAME category-var / options*;																
HBAR or VBAR														✓	✓	✓
HLINE or VLINE														✓	✓	✓
DOT														✓	✓	✓
Selected Options in Plot Statements																
GROUP /GROUP=*var*;	✓	✓	✓	✓	✓	✓	✓	✓	✓					✓	✓	✓
TRANSPARENCY /TRANSPARENCY=*value* ;	✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓
MARKERS /MARKERS;		✓	✓	✓											✓	
NOMARKERS /NOMARKERS;							✓	✓	✓							
LEGENDLABEL /LEGENDLABEL= *'text-string'*;	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FILLATTRS /FILLATTRS= (*attribute=val*);						✓				✓		✓		✓		
LINEATTRS /LINEATTRS= (*attribute=val*);		✓	✓	✓	✓	✓	✓	✓	✓	✓			✓		✓	
MARKERATTRS /MARKERATTRS= (*attribute=val*);	✓	✓	✓	✓			✓	✓	✓						✓	✓

Table 2. SGPLOT plot statements and selected options.

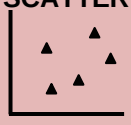
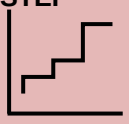
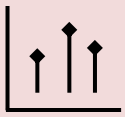
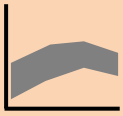
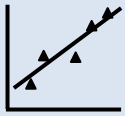
	SYNTAX	SELECTED OPTIONS	
SCATTER 	`SCATTER X=var Y=var/options;`	`DATALABEL=var`	Displays a label for each DATA point
SERIES 	`SERIES X=var Y=var/options;`	`BREAK`	Creates a break in the line for each missing value
		`CURVELABEL`	Labels the series curve using the Y variable label
STEP 	`STEP X=var Y=var/options;`	`BREAK`	Creates a break in the line for each missing value
		`CURVELABEL`	Labels the step curve using the Y variable label
NEEDLE 	`NEEDLE X=var Y=var/options;`	`BASELINE=val`	Specifies a numeric value on the Y axis for the baseline
VECTOR 	`VECTOR X=var Y=var/options;` (Note: VECTOR is new with SAS 9.2. Phase 2.)	`XORIGIN=val`	Specifies X coordinate for origin, either numeric value or numeric variable.
		`YORIGIN=val`	Specifies Y coordinate for origin, either numeric value or numeric variable.
BAND 	`BAND X=var UPPER=var LOWER=var/options;`	`FILL` `NOFILL`	Specifies if fill is visible or not
		`OUTLINE` `NOOUTLINE`	Specifies if outline is visible or not
REG 	`REG X=var Y=var/options;`	`ALPHA=val`	Specifies confidence level (default 0.05)
		`CLI`	Displays confidence limits for individual predicted values
		`CLM`	Displays confidence limits for mean predicted values
LOESS 	`LOESS X=var Y=var/options;`	`ALPHA=val`	Specifies confidence level (default 0.05)
		`CLM`	Displays confidence limits
		`INTERPOLATION=`	Specifies degree of interpolation: CUBIC (default) or LINEAR

Table 2 (continued). SGPLOT plot statements and selected options.

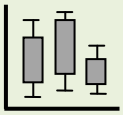
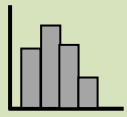
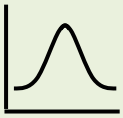
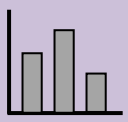
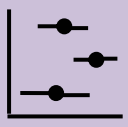
	SYNTAX	SELECTED OPTIONS	
PBSPLINE 	`PBSPLINE X=var Y=var/ options;`	`ALPHA=val`	Specifies confidence level (default 0.05)
		`CLI`	Displays confidence limits for individual predicted values
		`CLM`	Displays confidence limits for mean predicted values
ELLIPSE 	`ELLIPSE X=var Y=var/options;`	`ALPHA=val`	Specifies confidence level for the ellipse
		`TYPE=`	Specifies type of ellipse: MEAN or PREDICTED (default)
HBOX/VBOX 	`VBOX response-var/options;` `HBOX response-var/options;`	`CATEGORY=var`	A box plot is created for each value of the category variable
		`MISSING`	Creates box plot for missing values of category variable
HISTOGRAM 	`HISTOGRAM response-var/ options;`	`SHOWBINS`	Places tic mark at midpoint of bin
		`SCALE=`	Specifies scale for vertical axis: PERCENT (default), COUNT or PROPORTION
DENSITY 	`DENSITY response-var/ options;`	`SCALE=`	Specifies scale for vertical axis: DENSITY (default), PERCENT COUNT or PROPORTION
		`TYPE=`	Specifies type of density function: NORMAL (default) or KERNEL
HBAR/VBAR 	`VBAR category-var/options;` `HBAR category-var/options;`	`RESPONSE=var`	Specifies a numeric response variable for plot
		`STAT=`	Specifies statistic for axis of response variable (if specified): MEAN, or SUM (default)
		`BARWIDTH=num`	Specifies numeric value for width of bars (default 0.8)
HLINE/VLINE 	`VLINE category-var/options;` `HLINE category-var/options;`	`RESPONSE=var`	Specifies numeric response variable for plot
		`STAT=`	Specifies statistic for axis of response variable (if specified): MEAN, or SUM (default)
DOT 	`DOT category-var/options;`	`RESPONSE=var`	Specifies numeric response variable for plot
		`STAT=`	Specifies statistic for axis of response variable (if specified): MEAN, or SUM (default)
		`LIMITSTAT=`	Specifies statistic for limit lines (must use STAT=MEAN): CLM (default), STDDEV, or STDERR

Table 3. Selected optional statements with selected options.

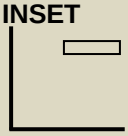
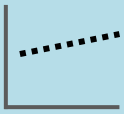
	SYNTAX	SELECTED OPTIONS	
REFLINE 	`REFLINE value1 value2 .../ options;`	`AXIS=`	Specifies axis for reference line: X, Y (default), X2, or Y2
		`LABEL=( )`	Creates labels for reference lines: (*'text1' 'text2' ...*)
		`LABELLOC=`	Specifies placement of label with respect to plot area: INSIDE (default) or OUTSIDE
		`LINEATTRS=( )`	Specifies attributes for reference line: (*attribute=value*)
XAXIS/YAXIS 	`XAXIS options;` `YAXIS options;`	`GRID`	Creates grid line at each tick on the axis
		`LABEL='text'`	Specifies a label for the axis
		`TYPE=`	Specifies type of axis: DISCRETE, LINEAR, LOG, or TIME
		`VALUES=( )`	Specifies values for tics on the axis: (*num1*,*num2*,…) or (*num1* TO *num2* BY *increment*)
INSET 	`INSET 'text1' 'text2' ... / options;`	`BORDER`	Creates a border around text box
		`POSITION=`	Specifies position of text box within plot area: BOTTOM, BOTTOMLEFT, BOTTOMRIGHT, LEFT, RIGHT, TOP, TOPLEFT, or TOPRIGHT

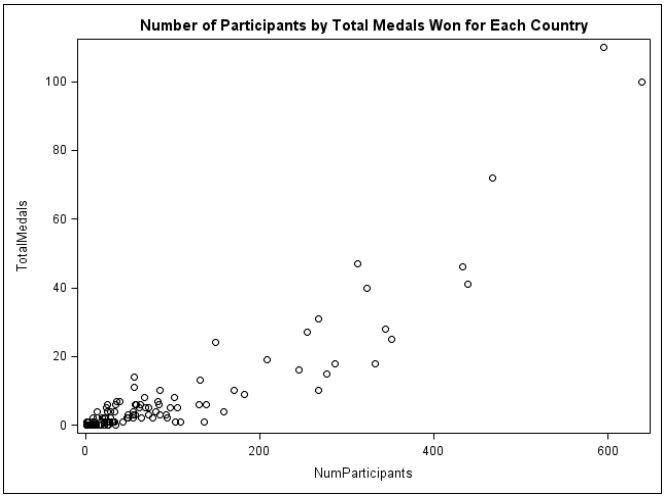
Table 4. The attribute options for plot statements.

	SYNTAX	ATTRIBUTES	
FILLATTRS 	/FILLATTRS=(*attribute=value*);	COLOR=	Specifies color for fill including: AQUA, BLACK, BLUE, FUCHSIA, GREEN, GRAY, LIME, MAROON, NAVY, OLIVE, PURPLE, RED, SILVER, TEAL, WHITE, and YELLOW
LINEATTRS 	/LINEATTRS=(*attribute=value*);	COLOR=	Specifies color for fill including: AQUA, BLACK, BLUE, FUCHSIA, GREEN, GRAY, LIME, MAROON, NAVY, OLIVE, PURPLE, RED, SILVER, TEAL, WHITE, and YELLOW
		PATTERN=	Specifies pattern for line including: SOLID, DASH, SHORTDASH, LONGDASH, DOT, DASHDASHDOT, and DASHDOTDOT
		THICKNESS=*val*	Specifies thickness of line. Value can include units: CM, IN, MM, PCT, PT, or PX (default)
MARKERATTRS 	/MARKEREATTRS= (*attribute=value*);	COLOR=	Specifies color for fill including: AQUA, BLACK, BLUE, FUCHSIA, GREEN, GRAY, LIME, MAROON, NAVY, OLIVE, PURPLE, RED, SILVER, TEAL, WHITE, and YELLOW
		SIZE=*val*	Specifies size of marker. Value can include units: CM, IN, MM, PCT, PT, or PX (default)
		SYMBOL=	Specifies symbol for marker including: CIRCLE, CIRCLEFILLED, DIAMOND, DIAMONDFILLED, PLUS, SQUARE, SQUAREFILLED, STAR, STARFILLED, TRIANGLE, and TRIANGLEFILLED

THE SGPANEL PROCEDURE

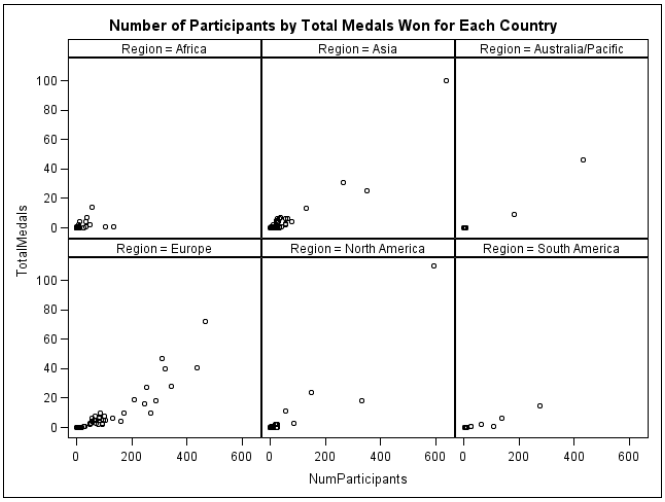
The SGPANEL procedure produces nearly all the same types of graphs as SGPLOT, but instead of displaying only one plot per image, SGPANEL can display several plots in a single image. A separate plot is produced for each level of the variable you specify in the PANELBY statement. To illustrate how SGPANEL works, here is the SGPLOT code to produce a scatter plot of the number of participants by the total medals won for each country.

```
*Scatter Plot using SGPLOT;
PROC SGPLOT DATA=Countries;
  SCATTER X = NumParticipants Y = TotalMedals;
  TITLE 'Number of Participants by Total Medals Won for Each Country';
RUN;
```



The syntax for SGPANEL is almost identical to SGPLOT, so it is easy to convert SGPLOT code to SGPANEL by making just a couple changes to your code. To produce a panel of plots, replace the SGPLOT keyword with SGPANEL, and add the PANELBY statement. The PANELBY statement must appear before any statements that create plots. Here, the PANELBY statement produces a separate plot for each value of REGION.

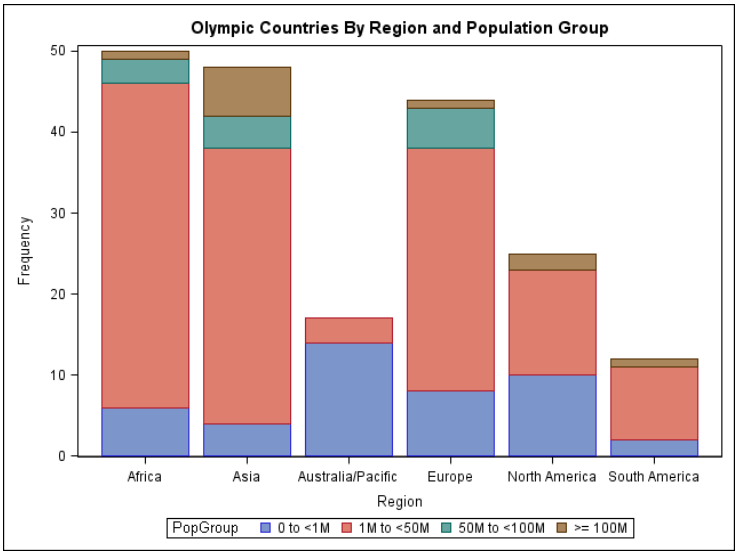
```
*Scatter plots using SGPANEL;
PROC SGPANEL DATA=Countries;
  PANELBY Region;
  SCATTER X = NumParticipants Y = TotalMedals;
  TITLE 'Number of Participants by Total Medals Won for Each Country';
RUN;
```



CHANGING THE ODS STYLE

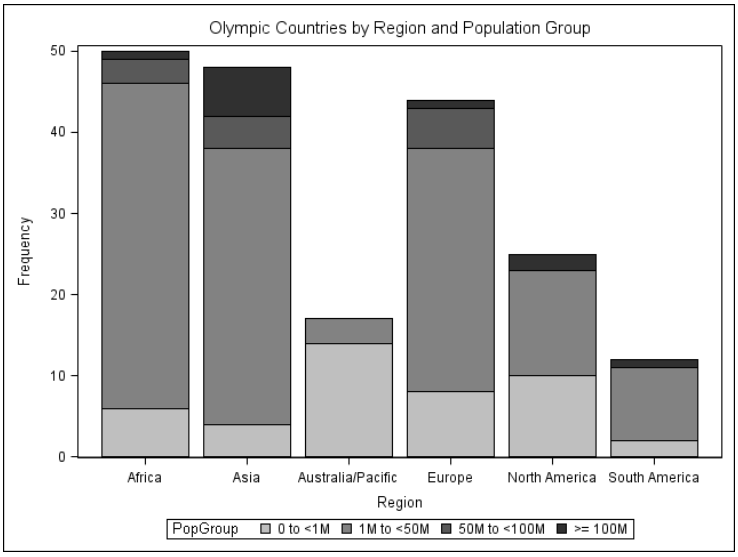
The following code creates a bar chart using the default ODS style template and destination. Since no destination is specified, the output will go to the LISTING destination. In the LISTING destination, tabular output is rendered as plain text so styles apply only to graphical output. The default style for the LISTING destination is named LISTING.

```
* Bar chart with default style;
PROC SGPLOT DATA = Countries;
   VBAR Region / GROUP = PopGroup;
   TITLE 'Olympic Countries by Region and Population Group';
RUN;
```



You can use the STYLE= option in an ODS destination statement to specify a style for your output including graphs. The following ODS LISTING statement changes the style to JOURNAL.

```
* Change ODS style template;
ODS LISTING STYLE = JOURNAL;
PROC SGPLOT DATA = Countries;
   VBAR Region / GROUP = PopGroup;
   TITLE 'Olympic Countries by Region and Population Group';
RUN;
```



You can use the same style templates for graphs as you do for tabular output. However, some styles are better suited for statistical graphs than others. The following table lists styles that are recommended for graphical results.

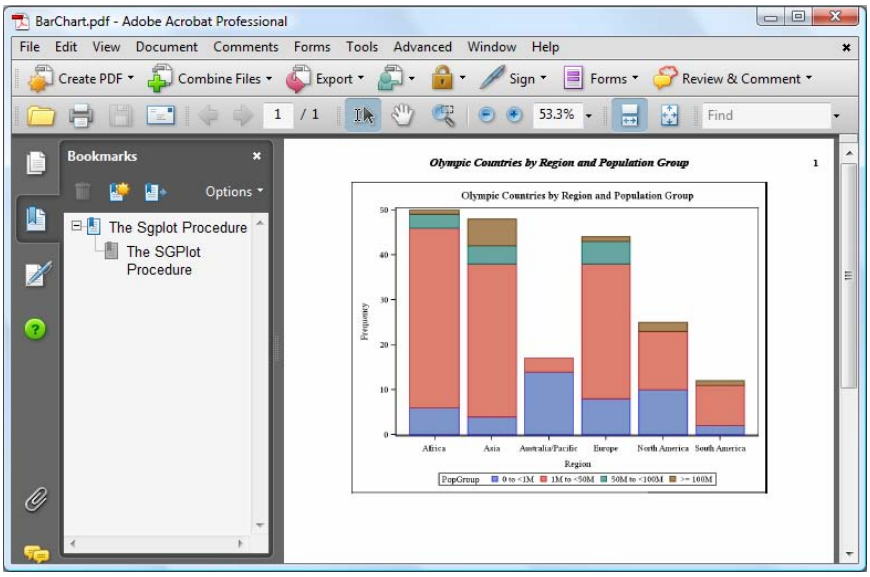
Desired Output	Style Template	Default for Destination
Color	ANALYSIS	
	DEFAULT	HTML
	LISTING	LISTING (graphs only)
	PRINTER	PRINTER, PDF, PS
	RTF	RTF
	STATISTICAL	
Gray scale	JOURNAL	
Black and white	JOURNAL2	

Every destination has a default style associated with it. This means that if you change the destination for a graph, its appearance may change too.

CHANGING THE ODS DESTINATION

You can send ODS Graphics output to any ODS destination, and you do it in the same way that you would for tabular output, using ODS statements for that destination. These statements send a bar chart to the PDF destination.

```
* Send graph to PDF destination;
ODS PDF FILE = 'c:\MyPDFFiles\BarChart.pdf';
PROC SGPLOT DATA = Countries;
   VBAR Region / GROUP = PopGroup;
   TITLE 'Olympic Countries by Region and Population Group';
RUN;
ODS PDF CLOSE;
```



SAVING ODS GRAPHICS OUTPUT AND ACCESSING INDIVIDUAL GRAPHS

For most destinations (including RTF and PDF), graphs are integrated with tabular output into a single file. For these destinations, you can use the FILE= option to tell SAS where to save your output. This statement would create PDF output and save it in file named Report.pdf in a folder named MyPDFFiles.

```
ODS PDF FILE = 'c:\MyPDFFiles\Report.pdf';
```

However, when you send output to the LISTING or HTML destinations, graphs are saved in individual files separate from tabular output. By default these files are saved in the current SAS working directory. (In the SAS windowing environment, the path of the current SAS working directory appears in the lower-right corner of the SAS window.) For the LISTING and HTML destinations, you can use the GPATH= option to tell SAS where to save individual graphs. This statement would save all graphs sent to the HTML destination in a folder named MyGraphs.

```
ODS HTML GPATH = 'c:\MyGraphs';
```

For the HTML destination, these individual graph files are linked to the tabular output so that they appear integrated when viewed in a Web browser. Therefore, when you send graphical output to the HTML destination, you must be careful to save those files in directories where they can be found by anyone who displays the HTML file in a Web browser.

If you are writing a paper or creating a presentation, you may need to access individual graphs. For the HTML, PDF, and RTF destinations in the Windows operating environment, you can simply copy and paste images when you view them in SAS. You cannot copy and paste images from the LISTING destination, but since the individual graphs are saved in separate files, you can open or insert those files as you would any standard image file.

SPECIFYING PROPERTIES OF IMAGES

Using the ODS GRAPHICS statement, you can control many aspects of your graphics files including the image format, name, height, and width. LISTING is the best destination for this since it offers the most image formats, and saves images in separate files. Here are a few of the options you can specify in the ODS GRAPHICS statement:

Option	Description	Example
IMAGEFMT=	specifies image format	`IMAGEFMT = JPEG`
IMAGENAME=	specifies base filename for image	`IMAGENAME = 'NewGraph'`
HEIGHT=	specifies height	`HEIGHT = 4IN`
WIDTH=	specifies width	`WIDTH = 8CM`
RESET	resets any preceding options to default values	`RESET`

Image formats supported for the LISTING destination include PNG (the default), BMP, GIF, JPEG, JPG, PDF, PS, TIFF, and many others.

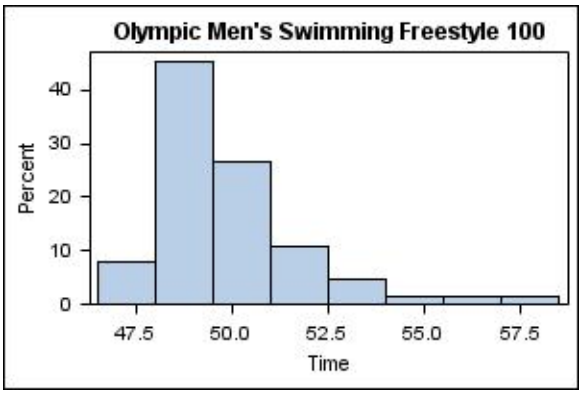
The default name for an image file is the name of its ODS output object. (You can use ODS TRACE statements to find the name of any ODS output object.) If you create more than one image file, SAS will append numerals to the end of the image name. For example, if you specified an image name of Olympic, then the files would be named Olympic, Olympic1, Olympic2, Olympic3, and so on. If you rerun your code, SAS will, by default, continue counting so that your graphics files will not be overwritten. If you want to start at zero each time you run your code, then specify the RESET option before the IMAGENAME= option.

In most cases, the default size for graphs is 640 pixels wide by 480 pixels high. If you specify only one dimension (width but not height, or vice versa), then SAS will adjust the other dimension to maintain a default aspect ratio of 4/3. You can specify width and height in these units: CM, IN, MM, PT, or PX.

The following code creates a graph in JPEG format that is 2 inches by 3 inches. The file will be named Final.jpeg and will be saved in a folder named MyGraphs.

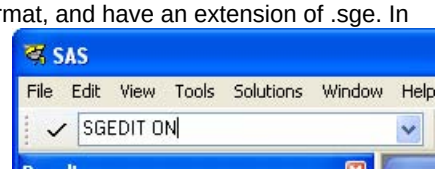
```
ODS GRAPHICS / RESET IMAGENAME = 'Final' IMAGEFMT =JPEG
   HEIGHT = 2in WIDTH = 3in;
ODS LISTING GPATH = 'c:\MyGraphs' ;

* Histograms;
PROC SGPLOT DATA = Freestyle;
   HISTOGRAM Time;
   TITLE "Olympic Men's Swimming Freestyle 100";
RUN;
```



MODIFYING GRAPHS IN THE ODS GRAPHICS EDITOR

You can use the ODS Graphics Editor to make one-time changes to graphs. The ODS Graphics Editor is available for all operating environments except z/OS and VMI. Before you can edit graphs, you must first tell SAS to create graphs in an editable form. Editable graphs are saved in ODS Graphics Editor file format, and have an extension of .sge. In SAS 9.2 Phase 1 and later, you can create editable graphs by clicking the Results window, and then submitting the **SGEDIT ON** command in the command box of the SAS windowing environment. If you want to turn off the creation of editable graphs (to save resources, for example), then submit the command, **SGEDIT OFF**.



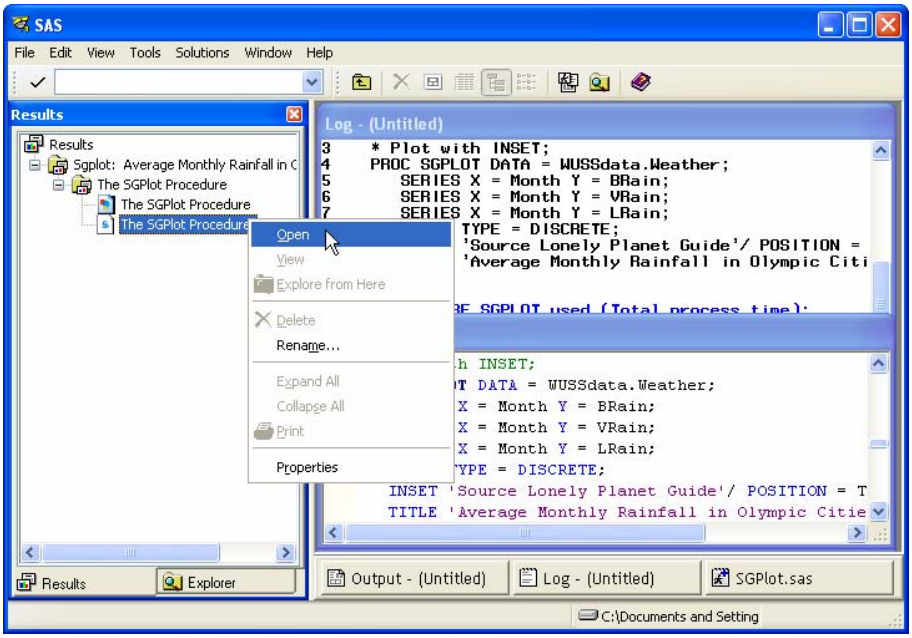
If you are using SAS 9.2 Phase 2, you can also create editable graphs in your program by adding this statement.

```
ODS LISTING SGE = ON;
```

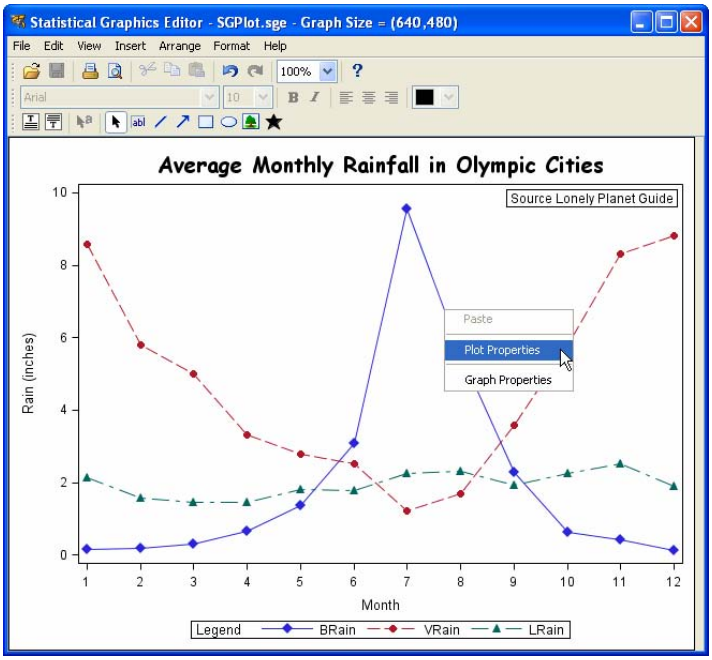
After turning on editable graphs, then submit the code that creates the graphs. If you want to stop creating editable graphs, then submit this statement

```
ODS LISTING SGE = OFF;
```

To open a graph in the ODS Graphics Editor in SAS 9.2 Phase 1, right-click the icon for the PNG image in the Results window, and select **Edit** from the pop-up menu. To open a graph in the ODS Graphics Editor in SAS 9.2 Phase 2, right-click the icon for the SGE image in the Results window, and select **Open** from the pop-up menu.



You can make many changes in the ODS Graphics Editor including changing the overall style, adding and editing titles and axis labels, and changing line types and markers. To make changes, use the menus and toolbar, or right-click individual parts of your plot (such as lines, titles, and axis labels) to open pop-up menus for that feature.



When you are done, you can use File ► Save As to save your graph in either PNG or SGE format. If you save it in SGE format, then you can return to edit the file again at a later time.

DOWNLOADING THE ODS GRAPHICS EDITOR

The ODS Graphics Editor is included with SAS/GRAPH software. Sometimes, however, the editor is not installed. If that is the case, then when you try to open an SGE image, you will see this message

```
This file does not have a program associated with it for performing this
action. Create an association in the Folder Options control panel.
```

If you see this message, then you need to install the editor. For the Windows or Linux operating environments, you can download the editor from the Downloads and Hot Fixes for SAS/GRAPH page on support.sas.com. The link is

```
http://www.sas.com/apps/demosdownloads/setupcat.jsp?cat=SAS%2FGRAPH+Software
```

CONCLUSIONS

ODS Graphics and the SGPLOT procedure introduce an exciting new way of producing high quality graphs using SAS. While PROC SGPLOT doesn't completely replace traditional SAS/GRAPH procedures, it does offer a wide variety of graphs using simple syntax. Because it is part of the Output Delivery System, you can use the same styles and destinations that you use for tabular output. A bonus of learning SGPLOT is that you can easily produce a panel of plots by converting your SGPLOT code to SGPANEL.

REFERENCES

Central Intelligence Agency (2007). “The World Factbook.” http://www.cia.gov/cia/publications/factbook/index.html.

Delwiche, L. D. Slaughter, S. J. (2008). *The Little SAS Book: A Primer, Fourth Edition*. SAS Institute, Cary, NC.

SAS Institute Inc. (2008). “SAS/GRAPH 9.2: Statistical Graphics Procedures Guide.” http://support.sas.com/documentation/onlinedoc/graph/index.html.

ABOUT THE AUTHORS

Lora Delwiche and Susan Slaughter are the authors of *The Little SAS Book: A Primer,* and *The Little SAS Book for Enterprise Guide* which are published by SAS Institute. The authors may be contacted at:

Susan J. Slaughter	Lora D. Delwiche
(530)756-8434	(530) 752-9321
susan@avocetsolutions.com	llddelwiche@ucdavis.edu

PRINT Procedure
Example 2: Selecting Variables to Print

Features:	PROC PRINT statement options: BLANKLINE DOUBLE STYLE VAR statement
Other features:	DATA step FOOTNOTE statement ODS HTML statement OPTIONS statement TITLE statement
Data set:	EXPREV
ODS destinations:	HTML LISTING

Details

This example demonstrates the following tasks:

- selects three variables for the reports
- uses variable labels as column headings
- double spaces between rows of the report in the LISTING output
- creates a report for the default HTML destination and the LISTING destination at the same time
- creates a stylized HTML report

Program: Creating an HTML Report

```
options obs=10;

ods listing;

proc print data=exprev;

   var country price sale_type;
   title 'Monthly Price Per Unit and Sale Type for Each Country';
   footnote '*prices in USD';
run;
```

Program Description

HTML is the default destination when SAS opens in the windowing environment.

Set the OBS= system option to process 10 observations.

```
options obs=10;
```

Open the LISTING destination. By default in the windowing environment, the HTML default is open. The ODS LISTING statement opens the LISTING destination in order to create HTML and LISTING output at the same time.

```
ods listing;
```

Print the output The VAR statement specifies the variables to print.

```
proc print data=exprev;
   var country price sale_type;
   title 'Monthly Price Per Unit and Sale Type for Each Country';
   footnote '*prices in USD';
run;
```

Output: HTML and LISTING

Selecting Variables: Default HTML Output

Monthly Price Per Unit and Sale Type for Each Country

Obs	Country	Price	Sale_Type
1	Antarctica	92.6	Internet
2	Puerto Rico	51.2	Catalog
3	Virgin Islands (U.S.)	31.1	In Store
4	Aruba	123.7	Catalog
5	Bahamas	113.4	Catalog
6	Bermuda	41.0	Catalog
7	Belize	146.4	In Store
8	British Virgin Islands	40.2	Catalog
9	Canada	11.8	Catalog
10	Cayman Islands	71.0	In Store

*prices in USD

Selecting Variables: LISTING Output

```
                  Monthly Price Per Unit and Sale Type for Each Country

                                                        Sale_
 Obs    Country                    Price      Type

   1    Antarctica                  92.6    Internet

   2    Puerto Rico                 51.2    Catalog

   3    Virgin Islands (U.S.)       31.1    In Store

   4    Aruba                      123.7    Catalog

   5    Bahamas                    113.4    Catalog

   6    Bermuda                     41.0    Catalog

   7    Belize                     146.4    In Store

   8    British Virgin Islands      40.2    Catalog

   9    Canada                      11.8    Catalog

  10    Cayman Islands              71.0    In Store


                          *prices in USD
```

Program: Creating an HTML Report with the STYLE and BLANKLINE Options

```
options obs=5;

ods html file='your_file_styles.html';

proc print data=exprev
   style(header)={fontstyle=italic color= green}
   style(obs)={backgroundcolor=#a8a44ff8a color=blue}
   blankline=(count= 1 style={backgroundcolor=cx456789});

   var country price sale_type;

   title 'Monthly Price Per Unit and Sale Type for Each Country';
   footnote '*prices in USD';
run;
```

Program Description

You can go a step further and add more formatting to your HTML output. The following example uses the STYLE option to add shading and spacing to your HTML report.

```
options obs=5;

ods html file='your_file_styles.html';
```

Create stylized HTML output. The first STYLE option specifies that the column headings are written in green italic font. The second STYLE option specifies that observation number column has a background color of the RGB color a8a44ff8a and a text color of blue. The BLANKLINE option specifies to add a blank line between each observation and use a background

color of the CMYK color cx456789. Because a style has not been defined for the OBSHEADER location, the Obs column heading in the output uses the default style color and not green.

```
proc print data=exprev
   style(header)={fontstyle=italic color= green}
   style(obs)={backgroundcolor=#a8a44ff8a color=blue}
   blankline=(count= 1 style={backgroundcolor=cx456789});

   var country price sale_type;

   title 'Monthly Price Per Unit and Sale Type for Each Country';
   footnote '*prices in USD';
run;
```

Output: HTML Output with Styles

Selecting Variables: HTML Output Using Styles

Monthly Price Per Unit and Sale Type for Each Country

Obs	Country	Price	Sale_Type
1	Antarctica	92.6	Internet
2	Puerto Rico	51.2	Catalog
3	Virgin Islands (U.S.)	31.1	In Store
4	Aruba	123.7	Catalog
5	Bahamas	113.4	Catalog

*prices in USD

Program: Creating a LISTING Report

```
options nodate pageno=1 linesize=80 pagesize=30 obs=10;

ods html close;
ods listing;

proc print data=exprev double;

   var country price sale_type;

   title 'Monthly Price Per Unit and Sale Type for Each Country';
   footnote '*prices in USD';
run;

ods listing close;
ods html;
```

Program Description

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. The PAGENO=

option specifies the starting page number. The LINESIZE= option specifies the output line length, and the PAGESIZE= option specifies the number of lines on an output page. The OBS= option specifies the number of observations to display.

```
options nodate pageno=1 linesize=80 pagesize=30 obs=10;
```

Close the HTML destination and open the LISTING destination. HTML is the default destination when you start SAS. To create only a LISTING report, you can close the HTML destination and open the LISTING destination.

```
ods html close;
ods listing;
```

Print the data set EXPREV. EXPREV contains information about a company's product order type and price per unit for two months. DOUBLE inserts a blank line between observations. The DOUBLE option has no effect on the HTML output.

```
proc print data=exprev double;
```

Select the variables to include in the report. The VAR statement creates columns for Country, Price, and Sale_Type, in that order.

```
    var country price sale_type;
```

Specify a title and a footnote. The TITLE statement specifies the title for the report. The FOOTNOTE statement specifies a footnote for the report.

```
    title 'Monthly Price Per Unit and Sale Type for Each Country';
    footnote '*prices in USD';
run;
```

Close the LISTING destination and reopen the HTML destination. When you close and reopen the HTML destination, SAS saves HTML output to the current directory and not the Work library.

```
ods listing close;
ods html;
```

Output: LISTING

By default, PROC PRINT identifies each observation by number under the column heading **`Obs.`**

Selecting Variables: LISTING Output

Monthly Price Per Unit and Sale Type for Each Country 1

Obs	Country	Price	Sale_ Type
1	Antarctica	92.6	Internet
2	Puerto Rico	51.2	Catalog
3	Virgin Islands (U.S.)	31.1	In Store
4	Aruba	123.7	Catalog
5	Bahamas	113.4	Catalog
6	Bermuda	41.0	Catalog
7	Belize	146.4	In Store
8	British Virgin Islands	40.2	Catalog
9	Canada	11.8	Catalog
10	Cayman Islands	71.0	In Store

*prices in USD

CONTENTS Procedure
Example 1: Describing a SAS Data Set

Features:	PROC CONTENTS statement options: DATA= OUT=
Other features:	OPTIONS statement TITLE statement

Details

This example shows the output from the CONTENTS procedure for the Group data set. The output shows the modifications made to the Group data set in Modifying SAS Data Sets and the contents of the Grpout data set.

Program

```
options pagesize=40 linesize=80 nodate pageno=1;

LIBNAME health 'SAS-library';

proc datasets library=health nolist;
run;

proc contents data=health.group (read=green) out=health.grpout;
   title  'The Contents of the GROUP Data Set';
run;

proc contents data=health.grpout;
   title  'The Contents of the GRPOUT Data Set';
run;
```

Program Description

Set the system options. The PAGESIZE= option specifies the number of lines that compose a page of the SAS log and SAS output. The LINESIZE= option specifies the line size for the SAS log and for the SAS procedure output. The NODATE option specifies that the date and the time are not printed. The PAGENO= option specifies a beginning page number for the next page of output.

```
options pagesize=40 linesize=80 nodate pageno=1;
```

Set your libref.

```
LIBNAME health 'SAS-library';
```

Specify Health as the procedure input library, and suppress the directory listing.

```
proc datasets library=health nolist;
run;
```

Create the output data set Grpout from the data set Group. Specify Group as the data set to describe, give Read access to the Group data set, and create the output data set Grpout.

```
proc contents data=health.group (read=green) out=health.grpout;
   title  'The Contents of the GROUP Data Set';
run;
```

Display the contents of the Grpout data set.

```
proc contents data=health.grpout;
   title  'The Contents of the GRPOUT Data Set';
run;
```

Output Examples

Contents of the Group Data Set

The Contents of the GROUP Data Set

The DATASETS Procedure

Data Set Name	HEALTH.GROUP	**Observations**	148
Member Type	DATA	**Variables**	11
Engine	V9	**Indexes**	0
Created	09/03/2014 10:35:02	**Observation Length**	96
Last Modified	09/03/2014 10:35:02	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	NO
Label			
Data Representation	WINDOWS_32		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	8192
Number of Data Set Pages	3
First Data Page	1
Max Obs per Page	84
Obs in First Data Page	63
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\procdatasets\health\group.sas7bdat
Release Created	9.0401M3
Host Created	W32_7PRO
Owner Name	BUILTIN\Administrators
File Size	32KB
File Size (bytes)	32768

Alphabetic List of Variables and Attributes

#	Variable	Type	Len	Format	Informat
9	BIRTH	Num	8		
4	CITY	Char	15		
3	FNAME	Char	15		
10	HIRED	Num	8	DATE7.	DATE7.
11	HPHONE	Char	12		
1	IDNUM	Char	4		
7	JOBCODE	Char	4		
2	LNAME	Char	15		
8	SALARY	Num	8	COMMA8.	
6	SEX	Char	2		
5	STATE	Char	3		

Contents of the Grpout Data Set

The Contents of the GRPOUT Data Set

The CONTENTS Procedure

Data Set Name	HEALTH.GRPOUT	**Observations**	11
Member Type	DATA	**Variables**	41
Engine	V9	**Indexes**	0
Created	08/17/2016 09:21:07	**Observation Length**	888
Last Modified	08/17/2016 09:21:07	**Deleted Observations**	0
Protection		**Compressed**	NO
Data Set Type		**Sorted**	YES
Label			
Data Representation	WINDOWS_64		
Encoding	wlatin1 Western (Windows)		

Engine/Host Dependent Information	
Data Set Page Size	73728
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	82
Obs in First Data Page	11
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	c:\\procdatasets\health\grpout.sas7bdat
Release Created	9.0401M4
Host Created	X64_7PRO
Owner Name	BUILTIN\Administrators
File Size	144KB
File Size (bytes)	147456

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
32	CHARSET	Char	8		Host Character Set
33	COLLATE	Char	8		Collating Sequence
28	COMPRESS	Char	8		Compression Routine
20	CRDATE	Num	8	DATETIME16.	Create Date
22	DELOBS	Num	8		Deleted Observations in Data Set
36	ENCRYPT	Char	8		Encryption Routine
19	ENGINE	Char	8		Engine Name
27	FLAGS	Char	3		Update Flags (Protect Contribute Add)
10	FORMAT	Char	32		Variable Format
12	FORMATD	Num	8		Number of Format Decimals
11	FORMATL	Num	8		Format Length
38	GENMAX	Num	8		Maximum Number of Generations
40	GENNEXT	Num	8		Next Generation Number
39	GENNUM	Num	8		Generation Number
25	IDXCOUNT	Num	8		Number of Indexes for Data Set
23	IDXUSAGE	Char	9		Use of Variable in Indexes
13	INFORMAT	Char	32		Variable Informat
15	INFORMD	Num	8		Number of Informat Decimals
14	INFORML	Num	8		Informat Length
16	JUST	Num	8		Justification
9	LABEL	Char	256		Variable Label
7	LENGTH	Num	8		Variable Length

1	LIBNAME	Char	8		Library Name
3	MEMLABEL	Char	256		Data Set Label
2	MEMNAME	Char	32		Library Member Name
24	MEMTYPE	Char	8		Library Member Type
21	MODATE	Num	8	DATETIME16.	Last Modified Date
5	NAME	Char	32		Variable Name
18	NOBS	Num	8		Observations in Data Set
34	NODUPKEY	Char	3		Sort Option: No Duplicate Keys
35	NODUPREC	Char	3		Sort Option: No Duplicate Records
17	NPOS	Num	8		Position in Buffer
37	POINTOBS	Char	3		Point to Observations
26	PROTECT	Char	3		Password Protection (Read Write Alter)
29	REUSE	Char	3		Reuse Space
30	SORTED	Num	8		Sorted and/or Validated
31	SORTEDBY	Num	8		Position of Variable in Sortedby Clause
41	TRANSCOD	Char	3		Character Variables Transcoded
6	TYPE	Num	8		Variable Type
4	TYPEMEM	Char	8		Special Data Set Type (From TYPE=)
8	VARNUM	Num	8		Variable Number

Sort Information	
Sortedby	LIBNAME MEMNAME
Validated	YES
Character Set	ANSI

SORT Procedure
Example 1: Sorting by the Values of Multiple Variables

Features:	PROC SORT statement option: OUT= BY statement
Other features:	PROC PRINT

Details

This example does the following:

- sorts the observations by the values of two variables
- creates an output data set for the sorted observations
- prints the results

Program

```
data account;
   input Company $ 1-22 Debt 25-30 AccountNumber 33-36
         Town $ 39-51;
   datalines;
Paul's Pizza             83.00  1019  Apex
World Wide Electronics  119.95  1122  Garner
Strickland Industries   657.22  1675  Morrisville
Ice Cream Delight       299.98  2310  Holly Springs
Watson Tabor Travel      37.95  3131  Apex
Boyd & Sons Accounting  312.49  4762  Garner
Bob's Beds              119.95  4998  Morrisville
Tina's Pet Shop          37.95  5108  Apex
Elway Piano and Organ    65.79  5217  Garner
Tim's Burger Stand      119.95  6335  Holly Springs
Peter's Auto Parts       65.79  7288  Apex
Deluxe Hardware         467.12  8941  Garner
Pauline's Antiques      302.05  9112  Morrisville
Apex Catering            37.95  9923  Apex
;

proc sort data=account out=bytown;

   by town company;
run;

proc print data=bytown;

   var company town debt accountnumber;

   title  'Customers with Past-Due Accounts';
   title2 'Listed Alphabetically within Town';
run;
```

Program Description

Create the input data set ACCOUNT. ACCOUNT contains the name of each business that owes money, the amount of money that it owes on its account, the account number, and the town where the business is located.

```
data account;
   input Company $ 1-22 Debt 25-30 AccountNumber 33-36
         Town $ 39-51;
   datalines;
Paul's Pizza             83.00  1019  Apex
World Wide Electronics  119.95  1122  Garner
Strickland Industries   657.22  1675  Morrisville
Ice Cream Delight       299.98  2310  Holly Springs
Watson Tabor Travel      37.95  3131  Apex
Boyd & Sons Accounting  312.49  4762  Garner
Bob's Beds              119.95  4998  Morrisville
Tina's Pet Shop          37.95  5108  Apex
Elway Piano and Organ    65.79  5217  Garner
Tim's Burger Stand      119.95  6335  Holly Springs
Peter's Auto Parts       65.79  7288  Apex
Deluxe Hardware         467.12  8941  Garner
Pauline's Antiques      302.05  9112  Morrisville
Apex Catering            37.95  9923  Apex
;
```

Create the output data set BYTOWN. OUT= creates a new data set for the sorted observations.

```
proc sort data=account out=bytown;
```

Sort by two variables. The BY statement specifies that the observations should be first ordered alphabetically by town and then by company.

```
   by town company;
run;
```

Print the output data set BYTOWN. PROC PRINT prints the data set BYTOWN.

```
proc print data=bytown;
```

Specify the variables to be printed. The VAR statement specifies the variables to be printed and their column order in the output.

```
   var company town debt accountnumber;
```

Specify the titles.

```
   title  'Customers with Past-Due Accounts';
   title2 'Listed Alphabetically within Town';
run;
```

Output: HTML

Sorting by the Values of Multiple Variables

Customers with Past-Due Accounts
Listed Alphabetically within Town

Obs	Company	Town	Debt	AccountNumber
1	Apex Catering	Apex	37.95	9923
2	Paul's Pizza	Apex	83.00	1019
3	Peter's Auto Parts	Apex	65.79	7288
4	Tina's Pet Shop	Apex	37.95	5108
5	Watson Tabor Travel	Apex	37.95	3131
6	Boyd & Sons Accounting	Garner	312.49	4762
7	Deluxe Hardware	Garner	467.12	8941
8	Elway Piano and Organ	Garner	65.79	5217
9	World Wide Electronics	Garner	119.95	1122
10	Ice Cream Delight	Holly Springs	299.98	2310
11	Tim's Burger Stand	Holly Springs	119.95	6335
12	Bob's Beds	Morrisville	119.95	4998
13	Pauline's Antiques	Morrisville	302.05	9112
14	Strickland Industries	Morrisville	657.22	1675

Last updated: June 20, 2018

SORT Procedure
Example 2: Sorting in Descending Order

Features:	This example BY statement option: DESCENDING
Other features:	PROC PRINT
Data set:	Account

Details

This example does the following:

- sorts the observations by the values of three variables
- sorts one of the variables in descending order
- prints the results

Program

```
proc sort data=account out=sorted;
   by town descending debt accountnumber;
run;
proc print data=sorted;
   var company town debt accountnumber;
   title  'Customers with Past-Due Accounts';
   title2 'Listed by Town, Amount, Account Number';
run;
```

Program Description

Create the output data set SORTED. OUT= creates a new data set for the sorted observations.

```
proc sort data=account out=sorted;
```

Sort by three variables with one in descending order. The BY statement specifies that observations should be first ordered alphabetically by town, then by descending value of amount owed, then by ascending value of the account number.

```
   by town descending debt accountnumber;
run;
```

Print the output data set SORTED. PROC PRINT prints the data set SORTED.

```
proc print data=sorted;
```

Specify the variables to be printed. The VAR statement specifies the variables to be printed and their column order in the output.

```
var company town debt accountnumber;
```

Specify the titles.

```
title  'Customers with Past-Due Accounts';
title2 'Listed by Town, Amount, Account Number';
run;
```

Output: HTML

Note that sorting last by AccountNumber puts the businesses in Apex with a debt of $37.95 in order of account number.

Sorting in Descending Order

Customers with Past-Due Accounts
Listed by Town, Amount, Account Number

Obs	Company	Town	Debt	AccountNumber
1	Paul's Pizza	Apex	83.00	1019
2	Peter's Auto Parts	Apex	65.79	7288
3	Watson Tabor Travel	Apex	37.95	3131
4	Tina's Pet Shop	Apex	37.95	5108
5	Apex Catering	Apex	37.95	9923
6	Deluxe Hardware	Garner	467.12	8941
7	Boyd & Sons Accounting	Garner	312.49	4762
8	World Wide Electronics	Garner	119.95	1122
9	Elway Piano and Organ	Garner	65.79	5217
10	Ice Cream Delight	Holly Springs	299.98	2310
11	Tim's Burger Stand	Holly Springs	119.95	6335
12	Strickland Industries	Morrisville	657.22	1675
13	Pauline's Antiques	Morrisville	302.05	9112
14	Bob's Beds	Morrisville	119.95	4998

Last updated: June 20, 2018

SORT Procedure
Example 3: Maintaining the Relative Order of Observations in Each BY Group

Features:	PROC SORT statement option: EQUALS | NOEQUALS
Other features:	PROC PRINT

Details

This example does the following:

- sorts the observations by the value of the first variable
- maintains the relative order with the EQUALS option
- does not maintain the relative order with the NOEQUALS option

Program

```
data insurance;
   input YearsWorked 1 InsuranceID 3-5;
   datalines;
5 421
5 336
1 209
1 564
3 711
3 343
4 212
4 616
;

proc sort data=insurance out=byyears1 equals;

   by yearsworked;
run;

proc print data=byyears1;

   var yearsworked insuranceid;

   title 'Sort with EQUALS';
run;

proc sort data=insurance out=byyears2 noequals;

   by yearsworked;
run;

proc print data=byyears2;

   var yearsworked insuranceid;
```

```
   title 'Sort with NOEQUALS';
run;
```

Program Description

Create the input data set INSURANCE. INSURANCE contains the number of years worked by all insured employees and their insurance IDs.

```
data insurance;
   input YearsWorked 1 InsuranceID 3-5;
   datalines;
5 421
5 336
1 209
1 564
3 711
3 343
4 212
4 616
;
```

Create the output data set BYYEARS1 with the EQUALS option. OUT= creates a new data set for the sorted observations. The EQUALS option maintains the order of the observations relative to each other.

```
proc sort data=insurance out=byyears1 equals;
```

Sort by the first variable. The BY statement specifies that the observations should be ordered numerically by the number of years worked.

```
   by yearsworked;
run;
```

Print the output data set BYYEARS1. PROC PRINT prints the data set BYYEARS1.

```
proc print data=byyears1;
```

Specify the variables to be printed. The VAR statement specifies the variables to be printed and their column order in the output.

```
   var yearsworked insuranceid;
```

Specify the title.

```
   title 'Sort with EQUALS';
run;
```

Create the output data set BYYEARS2. OUT= creates a new data set for the sorted observations. The NOEQUALS option does not maintain the order of the observations relative to each other.

```
proc sort data=insurance out=byyears2 noequals;
```

Sort by the first variable. The BY statement specifies that the observations should be ordered numerically by the number

of years worked.

```
    by yearsworked;
run;
```

Print the output data set BYYEARS2. PROC PRINT prints the data set BYYEARS2.

```
proc print data=byyears2;
```

Specify the variables to be printed. The VAR statement specifies the variables to be printed and their column order in the output.

```
    var yearsworked insuranceid;
```

Specify the title.

```
    title 'Sort with NOEQUALS';
run;
```

Output: HTML

Note that sorting with the EQUALS option versus sorting with the NOEQUALS option causes a different sort order for the observations where YearsWorked=3.

Sorting with the EQUALS Option

Sort with EQUALS

Obs	YearsWorked	InsuranceID
1	1	209
2	1	564
3	3	711
4	3	343
5	4	212
6	4	616
7	5	421
8	5	336

Sorting with the NOEQUALS Option

Sort with NOEQUALS

Obs	YearsWorked	InsuranceID
1	1	209
2	1	564
3	3	343
4	3	711
5	4	212
6	4	616
7	5	421
8	5	336

Last updated: June 20, 2018

SORT Procedure
Example 4: Retaining the First Observation of Each BY Group

Features:	PROC SORT statement option: NODUPKEY BY statement
Other features:	PROC PRINT
Data set:	Account
Note:	The EQUALS option must be in effect to ensure that the first observation for each BY group is the one that is retained by the NODUPKEY option. The EQUALS option is the default. If the NOEQUALS option has been specified, then one observation for each BY group is retained by the NODUPKEY option, but not necessarily the first observation.

Details

In this example, PROC SORT creates an output data set that contains only the first observation of each BY group. The NODUPKEY option prevents an observation from being written to the output data set when its BY value is identical to the BY value of the last observation written to the output data set. The resulting report contains one observation for each town where the businesses are located.

Program

```
proc sort data=account out=towns nodupkey;

   by town;
run;

proc print data=towns;

   var town company debt accountnumber;

   title 'Towns of Customers with Past-Due Accounts';
run;
```

Program Description

Create the output data set TOWNS but include only the first observation of each BY group. NODUPKEY writes only the first observation of each BY group to the new data set TOWNS. If you use the VMS operating environment sort, then the observation that is written to the output data set is not always the first observation of the BY group.

```
proc sort data=account out=towns nodupkey;
```

Sort by one variable. The BY statement specifies that observations should be ordered by town.

```
   by town;
run;
```

Print the output data set TOWNS. PROC PRINT prints the data set TOWNS.

```
proc print data=towns;
```

Specify the variables to be printed. The VAR statement specifies the variables to be printed and their column order in the output.

```
    var town company debt accountnumber;
```

Specify the title.

```
    title 'Towns of Customers with Past-Due Accounts';
run;
```

Output: HTML

The output data set contains only four observations, one for each town in the input data set.

Retaining the First Observation of Each BY Group

Towns of Customers with Past-Due Accounts

Obs	Town	Company	Debt	AccountNumber
1	Apex	Paul's Pizza	83.00	1019
2	Garner	World Wide Electronics	119.95	1122
3	Holly Springs	Ice Cream Delight	299.98	2310
4	Morrisville	Strickland Industries	657.22	1675

Last updated: June 20, 2018