

Effective Graphics Made Simple Using SAS/GRAPH® SG Procedures

Dan Heath, SAS Institute Inc., Cary, NC

ABSTRACT

There are many types of graphics displays that you might need to create on a daily basis. In SAS® 9.2, SAS/GRAPH introduced a family of new procedures that enable you to create graphs quickly and efficiently. With very little coding effort, you can create effective and attractive graphics that can be as simple as scatter plots and bar charts, or as complex as multi-page classification panels.

The new statistical graphics (SG) family of procedures includes SGPLOT, SGPANEL, and SGSCATTER. The SGPLOT procedure creates single-celled graphs that can be constructed with a variety of plot and chart types. The SGPANEL procedure creates paneled graphs in which the paneling is driven by classification variables. The SGSCATTER procedure creates paneled scatter plots and matrices that have support for fitted lines, confidence bands, and computed ellipses. These three procedures are designed with a syntax that is powerful yet concise. This presentation provides examples that illustrate how you can use these procedures in your own work.

INTRODUCTION

“One graph is more effective than another if its quantitative information can be decoded more quickly or more easily by most observers” (Robbins 2005). The goal of graph creators should be to convey the information that is contained in the data as clearly and concisely as possible. Many factors can affect clarity, including poor choice of chart type, clutter in the chart, and poor choice of style attributes. In SAS 9.2, SAS/GRAPH introduced three new procedures that help users deal with these factors to create graphics that effectively communicate the necessary information.

A BIT OF HISTORY

In SAS 9.1.3®, SAS introduced a pre-production version of a new graphics system that was called ODS GRAPHICS, which is now production in SAS 9.2. This system uses the concept of template-based graphics. By using PROC TEMPLATE, procedure writers and SAS users can use this system to define graphics in the same manner that they can define tables. This system was designed with the principles of effective graphics in mind, including these features:

- A variety of plot and chart types to handle many data situations
- Graph layout capabilities that help eliminate the clutter that is found in many single-celled displays
- Styles designed to emphasize the data and minimize the supporting items in the graph

Procedure writers use this system to define graphs that procedures generate automatically in the process of an analysis. SAS users can use this system to create very sophisticated graphs using terminology familiar to statisticians and analysts. With its power and flexibility, the system can have a learning curve, and many users are not interested in learning a new system to program custom graphics. These users have certain kinds of graphs that they need to create regularly and with a minimal amount of learning and effort.

SAS recognized this need and through SAS/GRAPH has developed a new family of SG procedures that are built on top of the ODS GRAPHICS system. These procedures leverage the power and flexibility of the system to allow users to create effective graphics for their everyday needs, using a syntax that is clear and concise.

The first installment of this new family includes the following procedures:

- The SGPLOT procedure, which is used to create single-celled graphs that have overlay capabilities
- The SGPANEL procedure, which is used to create paneled graphs that are driven by classification variables
- The SGSCATTER procedure, which is used to create comparative scatter plot panels, and has the capability to overlay fits and confidences

BENEFITS OF THESE PROCEDURES

LESS CODING – AN ILLUSTRATION

Imagine you are a researcher who is analyzing data from an MPG study, and you want to create a graph to compare the variables from your data to look for interesting trends. Using device-based graphics in SAS 9.1.3, you could create a panel such as the one in Figure 1.

In SAS 9.1.3, there are several steps involved in creating the panel in Figure 1:

- 1) Calculate the loess fit for each plot using PROC LOESS
- 2) Determine the appropriate styling for the fonts and plot primitives
- 3) Generate each plot in the panel using PROC GPLOT
- 4) Combine the plots into a panel using PROC GREPLAY

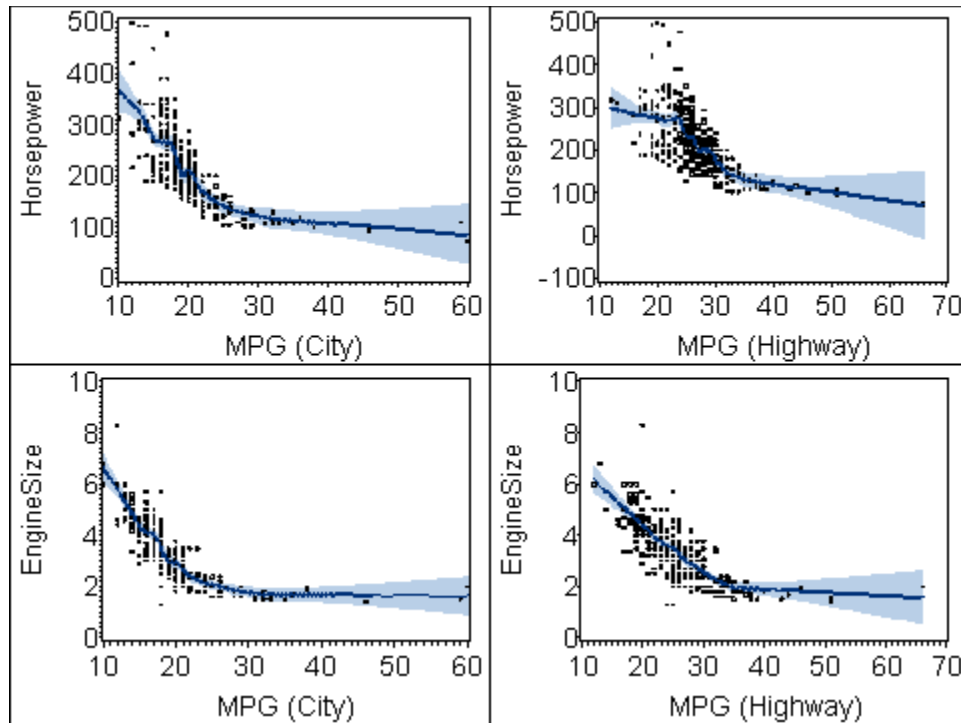


Figure 1: SAS 9.1.3 scatter plot panel with loess fits

As you can see, a program like this could take some time to develop. By the time the program is completed, you would have lost valuable time that could have been better used on the data analysis.

In SAS 9.2, you could create a panel like Figure 2 with just three lines of code:

```
proc sgscatter data=sashelp.cars;  
  plot (horsepower enginesize)*(mpg_city mpg_highway) /  
    markerattrs=(size=3) loess=(clm);  
run;
```

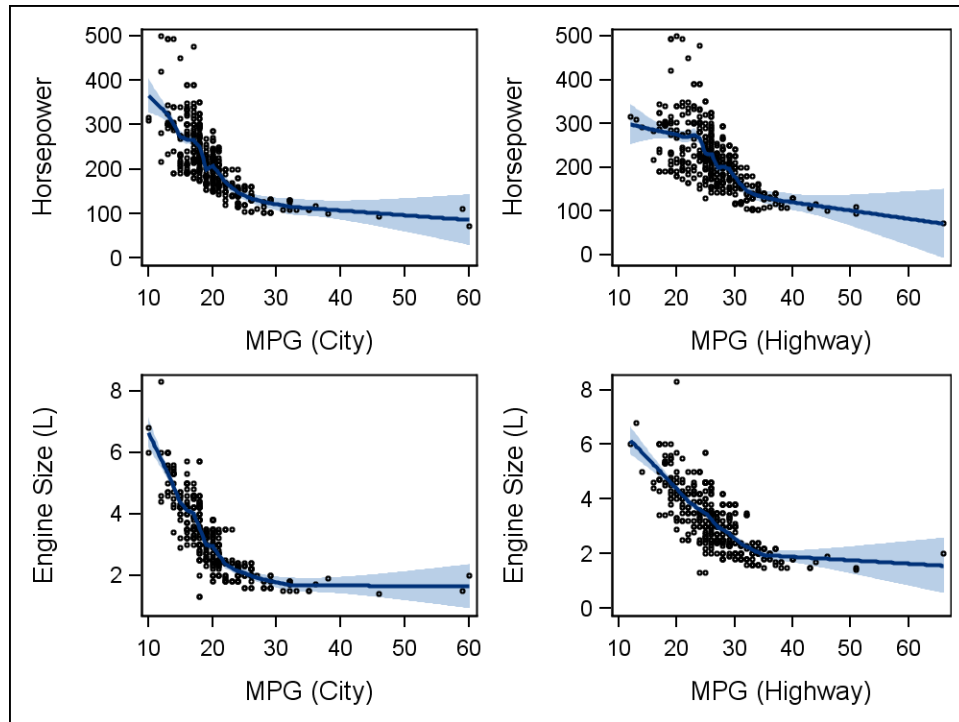


Figure 2: Plot panel using PROC SGSCATTER

CONSISTENT APPEARANCE FOR REPORTING

Over 50 SAS/STAT®, SAS/ETS®, SAS/HPF®, and SAS/QC® procedures use the ODS GRAPHICS system in 9.2 to automatically produce graphics that are based on the procedure analysis. Because the SG procedures use the same system, the graphics produced from the SG procedures will have an appearance that is consistent with the automatic ODS GRAPHICS output from the other procedures (Figures 3a and 3b).

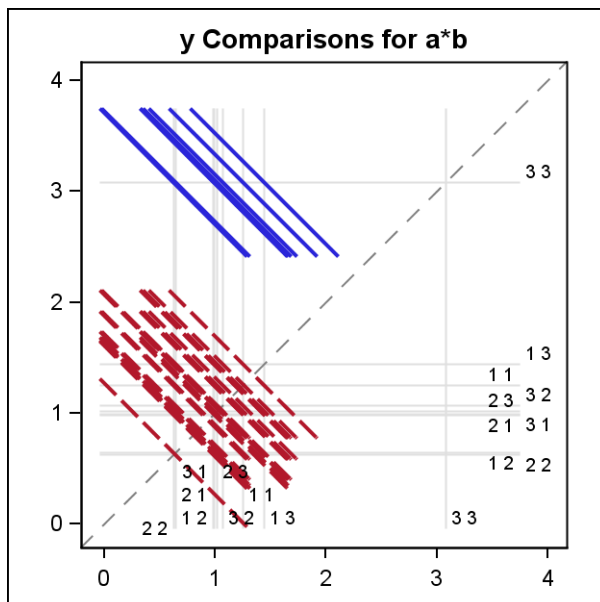


Figure 3a: A diffogram produced from PROC GLM

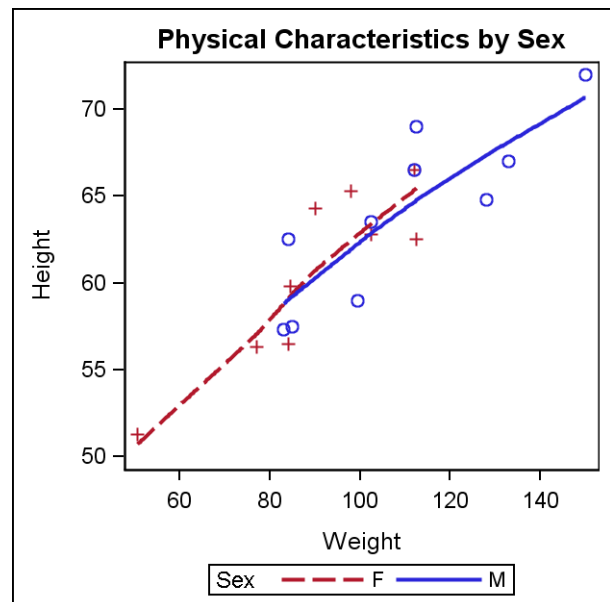


Figure 3b: A loess fit from PROC SGPLOT

Because these procedures are using a common graphical system, the automatic output from the statistical procedures can be combined with your custom graphs from the SG procedures to create a cohesive-looking report with a minimal amount of effort.

STATISTICAL STYLING

In SAS 9.2, SAS/GRAPH procedures use ODS styles for default graph appearance attributes. This functionality not only reduces your coding effort, but it also eliminates the need for you to determine effective colors and attributes. However, the SG procedures take this styling a step further. Because these procedures use the ODS GRAPHICS system, the procedures are able to use additional style elements in the ODS styles that define the appearance of different statistical features, such as box plots, fit lines, and confidence limits. In addition, you can access these style elements directly from your procedure code.

For example, if you calculated a custom fit for your data, you might use a SERIES plot to render the fit (Figure 4a):

```
proc sgplot data=sashelp.classfit noautolegend;  
  scatter x=height y=weight;  
  series x=height y=predict;  
run;
```

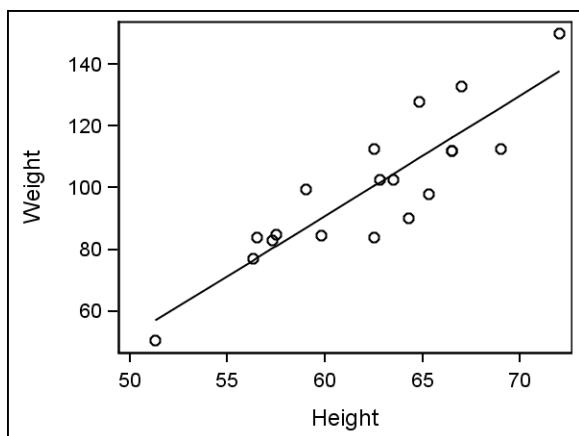


Figure 4a: Default styling for SERIES line

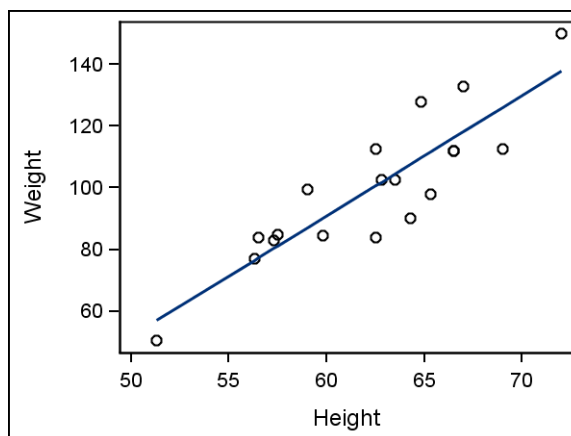


Figure 4b: SERIES line using the GraphFit style element

To improve the appearance of the SERIES line, you can assign the “GraphFit” style to the line attributes of the SERIES plot (Figure 4b):

```
proc sgplot data=sashelp.classfit noautolegend;  
  scatter x=height y=weight;  
  series x=height y=predict / lineattrs=GraphFit;  
run;
```

Style references have an additional benefit over literal style values: changing the ODS style for your output automatically changes the appearance of your graph output. For literal style values, you would need to examine your program code and make appropriate changes to match the rest of your output.

IMAGE QUALITY

Image quality is very important if you want to create effective graphics, particularly when you are reporting findings in a document. The ODS GRAPHICS system allows you to create high-resolution graphics without having to adjust any features in the graph. You can change this resolution by modifying the IMAGE_DPI setting on many of the ODS destination statements (for ODS PRINTER and ODS PDF, the DPI setting of the document sets the graphics resolution). The default image DPI for each destination is contained within the SAS registry. For document-based output, you generally want to create graphics with a resolution of 150 DPI or more; however, very high DPI settings can have an adverse impact on memory and performance. As a point of quality reference, all of the SG procedure output in this paper was created using 200 DPI.

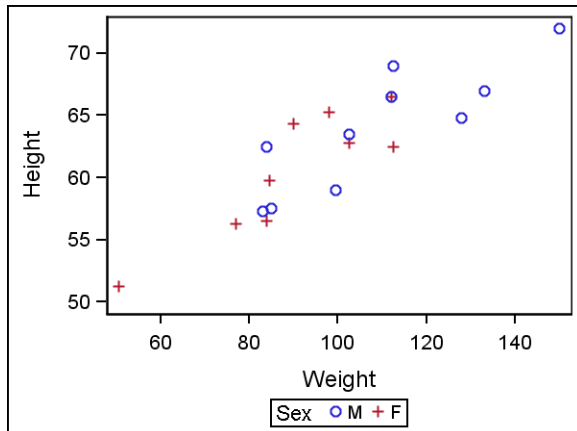
USING THE PROCEDURES

THE SGPLOT PROCEDURE

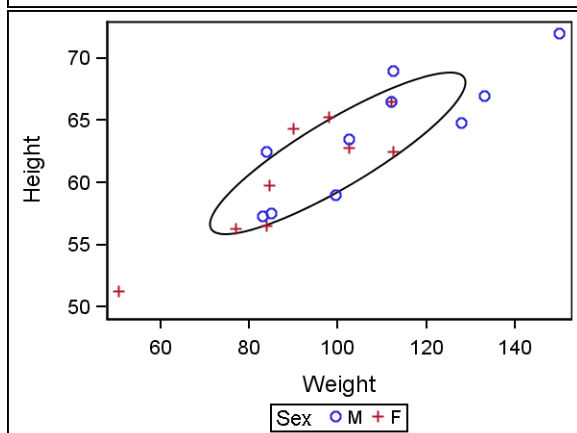
The SGPLOT procedure is designed to create single-celled graphs. A wide variety of plot and chart types are supported, including the following types:

- Basic plot: scatter, series, step, band, and needle
- Fits and Confidence: loess, regression, penalized B-spline, and computed ellipse
- Distribution: horizontal and vertical box plots, histograms, normal curves, and kernel density estimates
- Categorization: dot plots, horizontal and vertical bar charts, horizontal, and vertical line charts

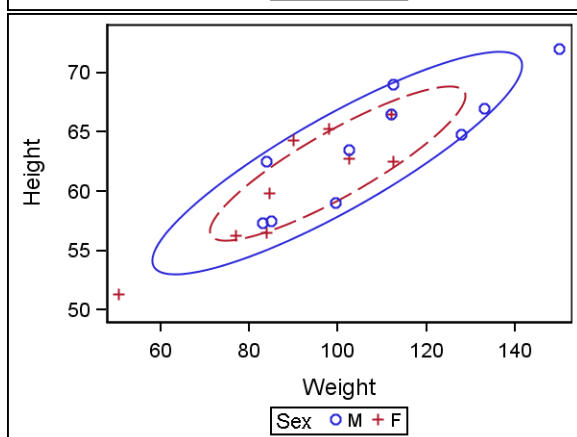
You can combine these plot types to create more complex plots as necessary. For example, Figure 5 shows how you can start with a simple scatter plot, then add multiple prediction ellipses just by adding additional statements.



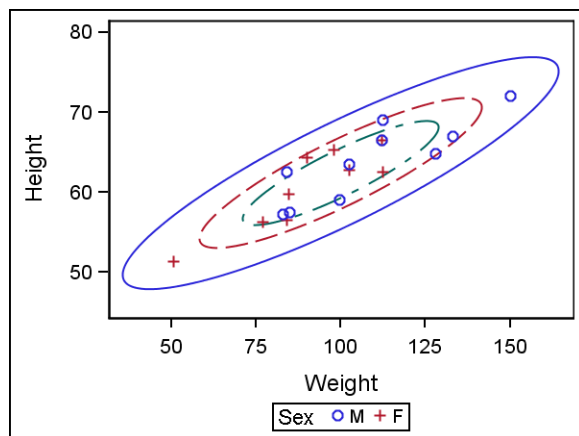
```
proc sgplot data=sashelp.class;
  scatter x=weight y=height / group=sex;
run;
```



```
proc sgplot data=sashelp.class;
  ellipse x=weight y=height / alpha=0.5;
  scatter x=weight y=height / group=sex;
run;
```



```
proc sgplot data=sashelp.class;
  ellipse x=weight y=height / alpha=0.25;
  ellipse x=weight y=height / alpha=0.5;
  scatter x=weight y=height / group=sex;
run;
```



```
proc sgplot data=sashelp.class;
  ellipse x=weight y=height / alpha=0.05;
  ellipse x=weight y=height / alpha=0.25;
  ellipse x=weight y=height / alpha=0.5;
  scatter x=weight y=height / group=sex;
run;
```

Figure 5: The SGPLOT procedure overlay example

Figure 5 would benefit from having an additional legend to describe the ellipses. The process for adding this legend is described later in Figure 7.

As you look down the table of plots, notice that the appearance of the graph changes as you add more ellipses. SGPLOT analyzes the plot statements to determine the best assignment of style attributes for the graph. For one ellipse in Figure 5, the default data style was determined to be the best look for the ellipse. For two or more ellipses, the procedure started to cycle the style attributes of the ellipses so that they can be identified separately.

Sometimes the procedure might not be able to determine the intent of the graph well enough to make good assignments, in which case each plot will use its default data style. In those cases, you can either use the CYCLEATTRS procedure option to force style attribute cycling, or you can assign your own attributes directly on the plot statements. Conversely, if the procedure chooses to cycle style attributes when you do not want to, you can specify NOCYCLEATTRS. The style attributes for each plot can also be specified directly on the plot statement.

Another point to consider in the example in Figure 5 is that the order of the plot statements is significant. The first plot statement is always drawn first, with each subsequent plot drawn on top of the previous plot. If the scatter plot was specified first, the lines of the ellipses would overwrite the scatter points in places. This problem would become more obvious if the ellipses were filled. However, plot statements in the SGPLOT procedure and the SG PANEL procedure do support transparency, which can be useful when filled regions and plots intersect.

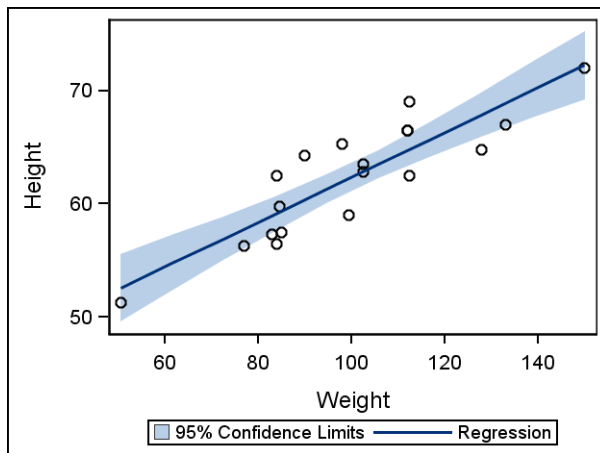
LEGEND CONTROL

The SGPLOT procedure gives you a lot of flexibility over the legends in your graphs. Not only can you control the location of a legend, but also the legend's content. You can also create additional legends as needed.

When you create a graph without any legend specifications, the SGPLOT procedure examines the graph request to determine if a legend should be displayed; and, if so, what plot information should be contained in that legend. That information falls into two categories:

- The plots that should be included in the legend
- The best label for each plot, including relevant statistical information if available

In Figure 6, we have a basic regression fit. The procedure determined that a legend was necessary to identify the fit and the limit feature. Notice that in this case the scatter plot has been excluded from the legend. The central focus of the plot was determined to be the regression fit; therefore, the scatter plot was not added to the legend. Also notice the legend labels have been customized to report the type of fit automatically, as well as the limit type and confidence level. These legend labels may be overridden by specifying a custom legend label on the plot statements.



```
proc sgplot data=sashelp.class;
  reg x=weight y=height / clm;
run;
```

Figure 6: Regression fit with automatic legend

There could be times when you want to create a custom legend, either because you want to control the content of a legend or because you want to have multiple legends. A legend is defined using the KEYLEGEND statement. Creating a legend with customized content suppresses the automatic legend. Alternatively, the automatic legend can be suppressed using the NOAUTOLEGEND procedure option.

Referring again to the example in Figure 5, the automatic legend facility determined that the grouped plot was the feature that needed to be shown in the legend. However, the prediction ellipses also needed to be identified to help make the plot more meaningful. Therefore, the best approach is to create two legends: one for the grouped scatter plot and one for the three ellipses (Figure 7).

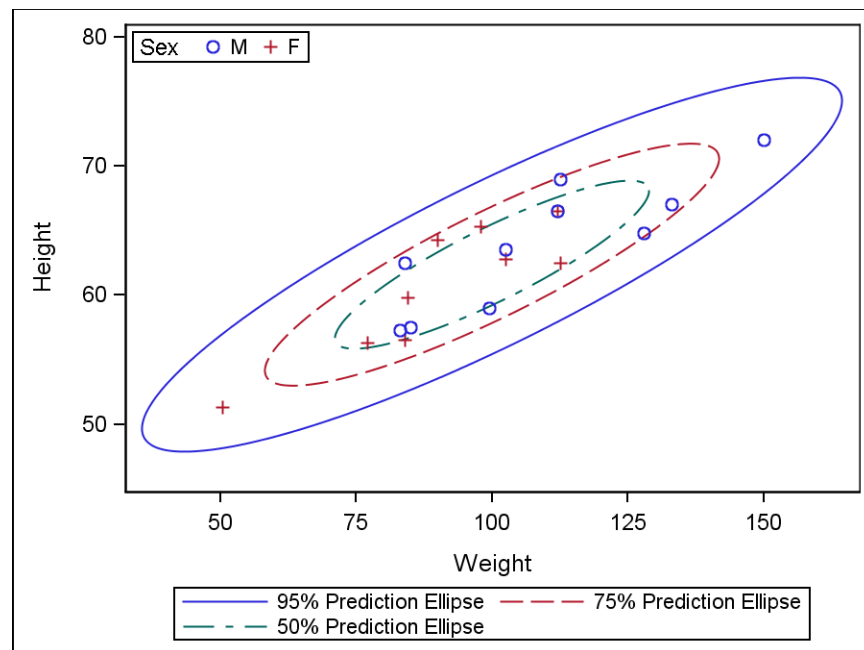


Figure 7: Plot with custom legends

```
proc sgplot data=sashelp.class;
  ellipse x=weight y=height / alpha=0.05 name="e1";
  ellipse x=weight y=height / alpha=0.25 name="e2";
  ellipse x=weight y=height / alpha=0.5 name="e3";
  scatter x=weight y=height / group=sex name="s1";
  keylegend "e1" "e2" "e3";
  keylegend "s1" / location=inside position=topleft title="Sex";
run;
```

For a custom legend, each plot that you want to include in the legend must be named using the NAME option. This plot name is also listed on the KEYLEGEND statement to have the plot's content included in the legend. The order of this list is significant: the first name in the list is included first, followed by each name in sequence. You can use this feature to control the ordering of the plots in your legend.

THE SGPANEL PROCEDURE

The SGPANEL procedure is designed to produce paneled graphs based on classification variables. As with the SGPLOT procedure, the SGPANEL procedure allows you to combine a variety of plot and chart types into more elaborate overlays. The SGPANEL procedure supports the following plot types:

- Basic plot: scatter, series, step, band, and needle
- Fits and Confidence: loess, regression, and penalized B-spline
- Distribution: horizontal and vertical box plots, histograms, normal curves, and kernel density estimates
- Categorization: dot plots, horizontal and vertical bar charts, and horizontal and vertical line charts

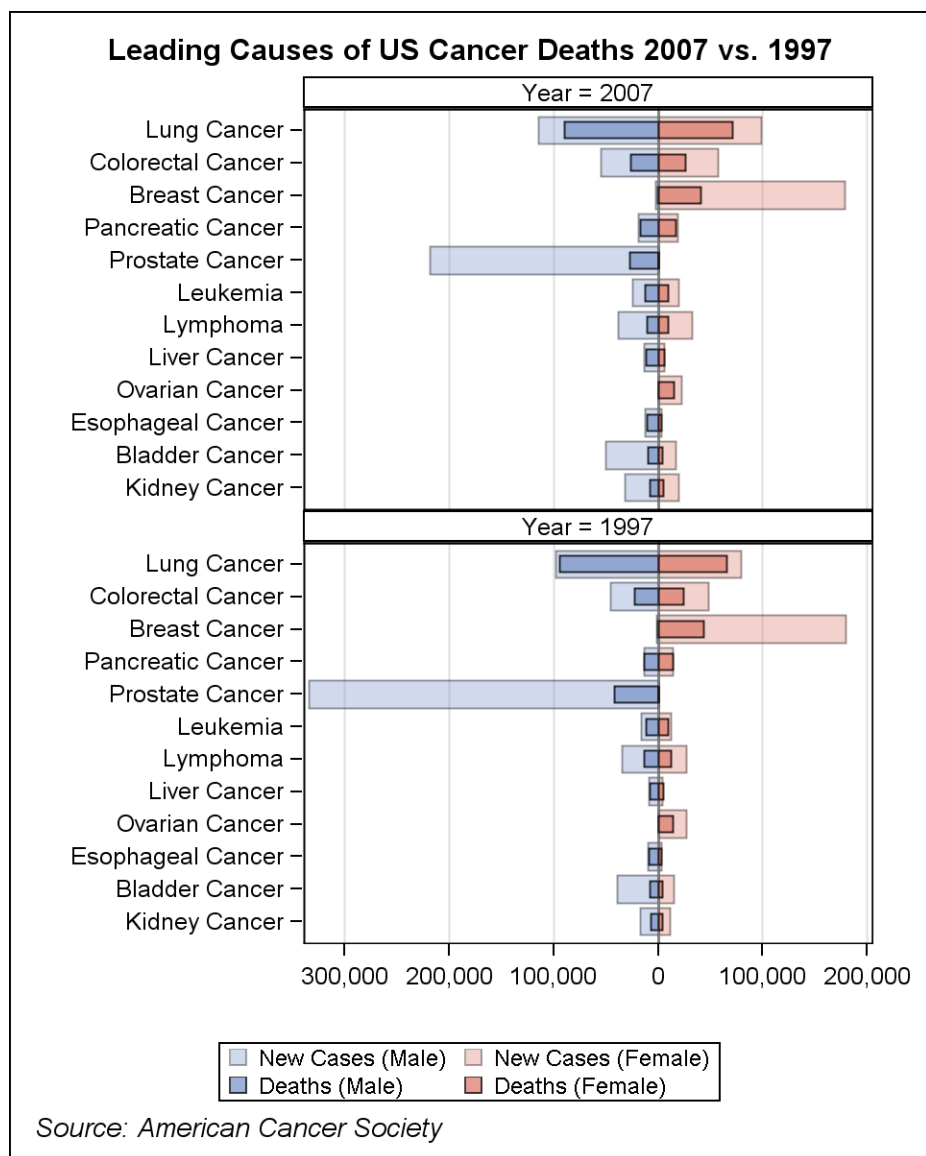


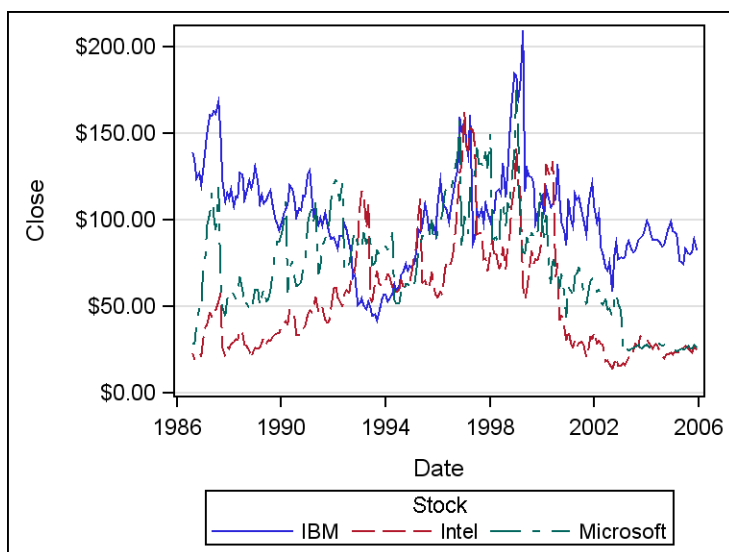
Figure 8: Butterfly panel using the SGPANEL procedure

The SGPLOT and SGPANEL procedures are designed to be syntactically similar, enabling you to quickly take a single-celled plot and expand it into a classification panel that uses higher-level classifiers. For example, Figure 8 illustrates a butterfly panel that was created using the SGPANEL procedure. The butterfly chart was originally

created by using the SGPLOT procedure, but the chart was easily converted into a panel by using the SGPANEL procedure. All that was required to create the graph in Figure 8 was to change the name of the procedure from SGPLOT to SGPANEL, add a PANELBY statement, and change the name of the axis statements from XAXIS/YAXIS to COLAXIS/ROWAXIS, respectively.

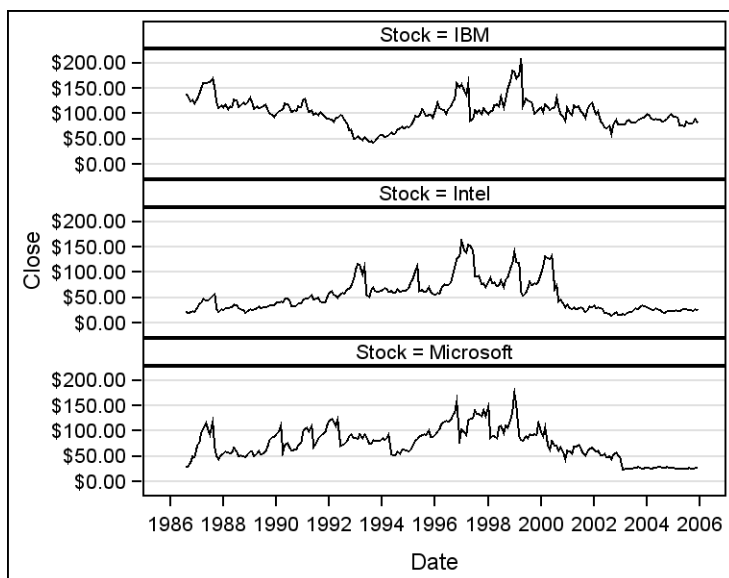
For more details about this example, as well as other interesting examples that use the SG procedures, please see Susan Schwartz's SAS Global Forum 2008 poster/paper, "Butterflies, Heat Maps and More: Explore the New Power of SAS/GRAPH."

You can use the SGPANEL procedure to reduce clutter in a single-celled graph, which makes the data easier to compare and makes the graph more effective. For example, suppose you have data for three stocks and you want to compare their closing prices over time. You might start by overlaying the closing prices for all three stocks in the same cell (Figure 9), but you might find it difficult to discern the trends for each stock. You can create a better display by using the SGPANEL procedure to separate the series into multiple cells, but combine them in the same panel. The trends are easier to see and compare (Figure 10). This technique is advocated by Robbins (2005).



```
proc sgplot data=sashelp.stocks;
  yaxis grid;
  series x=date y=close /
    group=stock;
run;
```

Figure 9: Overlaid series plots cause clutter.



```
proc sgpanel data=sashelp.stocks;
  panelby stock / columns=1;
  rowaxis grid;
  series x=date y=close;
run;
```

Figure 10: The SGPANEL procedure separates the series into multiple cells.

THE PANELBY STATEMENT

The PANELBY statement is the key statement in the SGPPANEL procedure. This required statement is where you specify the classifiers that you want to use to create the panel, and where you specify the layout of the panel.

The number of classifier variables that is allowed in the statement depends on the type of layout that you request. If you are using the *panel* layout (the default), there is no limit to the number of classifier variables that you can specify. If you are using the *lattice* layout, you must specify two classifier variables, one for the columns and one for the rows.

When creating effective graphics using the SGPPANEL procedure, how you specify the classifier variables and which type of layout you use are important considerations. As stated earlier, the *panel* layout enables you to specify as many classifiers as you want, but an effective display should typically have only one or two PANELBY variables. There are two reasons for this:

- Each variable specified on the PANELBY statement creates a header entry above the cell, which takes away space that could be better used by the plot in the cell.
- For a large number of PANELBY variable crossings, a panel might contain a number of redundant values that would be better extracted to a page-level classification.

Therefore, for a large number of classifiers, it is typically better to use a BY statement with a PANELBY statement to produce the graph. One exception to note is when aggregating formats are assigned to your classifier variables. If two or more variables are on the BY statement, and aggregating formats are used on any of those variables, you could lose the aggregation effect. Variables on the PANELBY statement always honor the aggregation, regardless of the number of variables. Therefore, in those situations, you might want to put all of the variables on the PANELBY statement. For more on aggregating formats, see Figures 13 and 14 later in the paper.

Here is an example of a case in which you could benefit by using a BY statement. You have a data set with two years of actual and predicted sales data, and you want to create a panel to see the results by year and by quarter. You might initially create a program similar to the following:

```
title1 "Product Sales";
proc sgppanel data=sashelp.prdsale;
  panelby year quarter;
  rowaxis label="Sales";
  vbar product / response=predict
               transparency=0.3;
  vbar product / response=actual barwidth=0.5
               transparency=0.3;
run;
```

Your output for 1993 would resemble the example in Figure 11. Notice that the year is redundant information in the graph. The focus of the graph should be on the quarters for that year.

Figure 12 contains a better display of the data. In this graph, the *year* variable has been moved to the BY statement so that the year is shown in the BY line of the graph, and the quarter is shown in the panel:

```
title1 "Product Sales";
proc sgppanel data=sashelp.prdsale;
  by year;
  panelby quarter;
  rowaxis label="Sales";
  vbar product / response=predict
               transparency=0.3;
  vbar product / response=actual barwidth=0.5
               transparency=0.3;
run;
```

If you run the Figure 11 and 12 examples, you find that the procedure produces two graphs with a 2x2 layout without any user specification required. If neither ROWS nor COLUMNS is specified on the PANELBY statement, the SGPPANEL procedure automatically determines a layout size and the number of graphs to generate based on the classifier crossings. The layout algorithm minimizes the number of empty cells in the last graph while maximizing the space for each plot in a cell. When one of the ROWS or COLUMNS options is specified, this same algorithm is also used to determine the value of the unspecified option. If both options are specified, the procedure uses both option values for the layout and generates graphs until all of the crossings are exhausted.

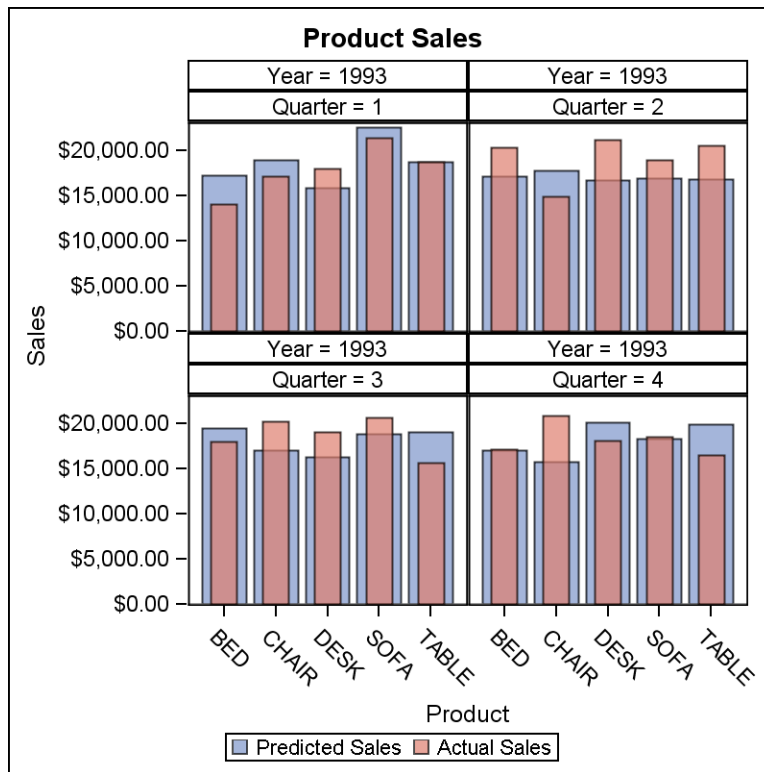


Figure 11: Using *year* and *quarter* in the PANELBY statement

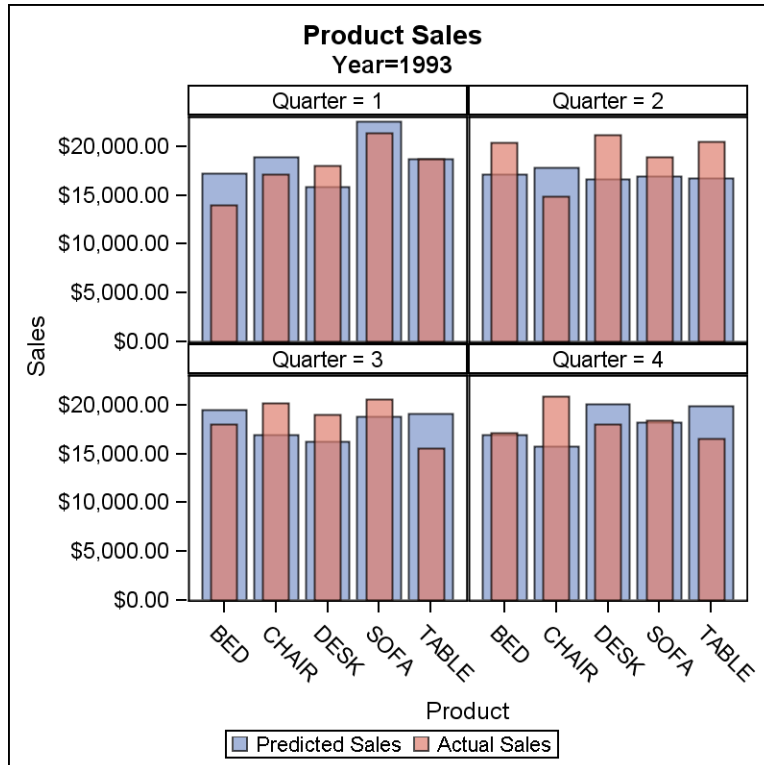


Figure 12: Using *year* in the BY statement and *quarter* in the PANELBY statement

Another consideration is the type of layout to use. Consider an example of a case in which you are comparing the cholesterol levels between males and females who have been diagnosed with coronary heart disease in a heart study (Figure 13):

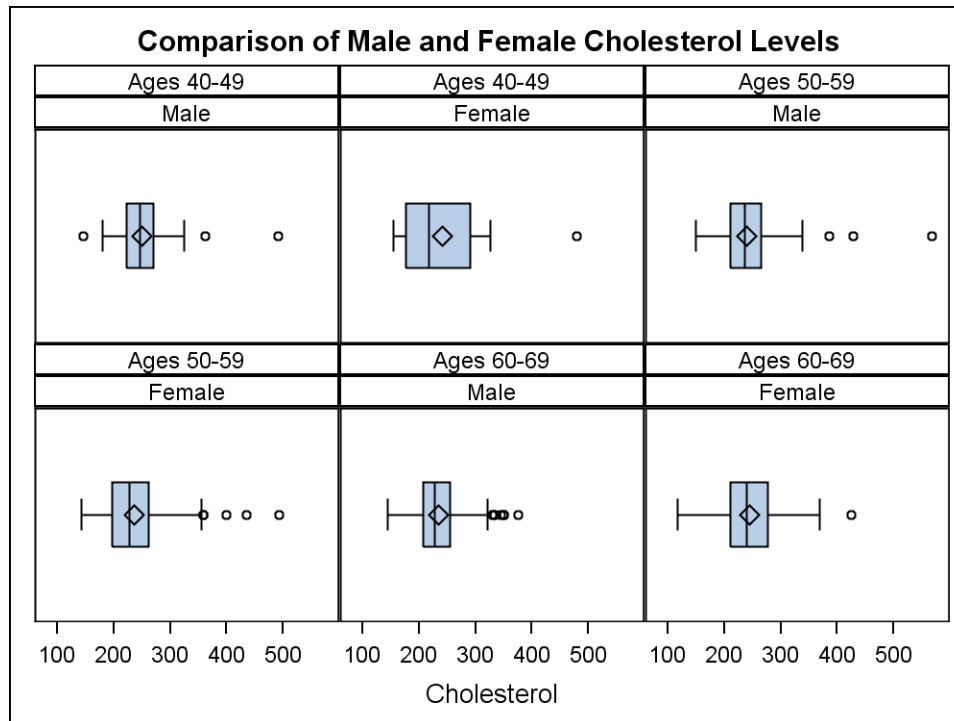


Figure 13: Horizontal box plot of cholesterol levels

```
proc format;
value ageint
    40-49 = "Ages 40-49"
    50-59 = "Ages 50-59"
    60-69 = "Ages 60-69";
run;

title1 "Comparison of Male and Female Cholesterol Levels";
proc sgpanel data=sashelp.heart;
    format AgeCHDdiag ageint.;
    where AgeCHDdiag>=40 and AgeCHDdiag<=69;
    panelby AgeCHDdiag sex / novarname;
    hbox cholesterol;
run;
```

Notice that the NOVARNAME option removes the variable name or the label from the header information. This option is useful when the classifier values are self-explanatory.

Although this display shows the information correctly, it is not as effective as it could be because you cannot easily compare the male and female cholesterol levels for each age group. A better display is created using the *lattice* layout (Figure 14):

```
proc format;
value ageint
    40-49 = "Ages 40-49"
    50-59 = "Ages 50-59"
    60-69 = "Ages 60-69";
run;
```

```

title1 "Comparison of Male and Female Cholesterol Levels";
proc sgpanel data=sashelp.heart;
  format AgeCHDdiag ageint.;
  where AgeCHDdiag>=40 and AgeCHDdiag<=69;
  panelby AgeCHDdiag sex / novarname layout=lattice;
  hbox cholesterol;
run;

```

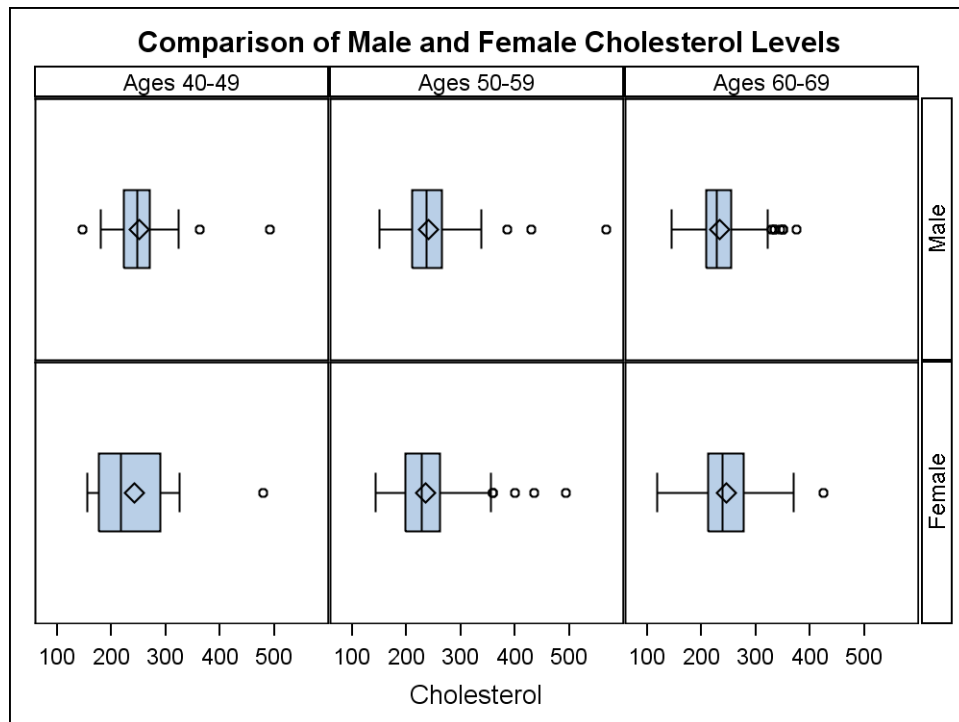


Figure 14: Horizontal box plot using a *lattice* layout

The first PANELBY variable is used for the column values and the second variable for the row values. In this display, you can look down the columns to make the gender comparison by age group.

Notice also in Figures 13 and 14 how the age ranges are defined. In this first version of the SG PANEL procedure, all classifier values are treated discretely, including numeric values, but the procedure honors the format aggregation. Therefore, you can create user-defined formats to *bucket* your numeric data into the desired ranges.

THE SGSCATTER PROCEDURE

The SGSCATTER procedure is designed to create panels of scatter plots and scatter plot matrices. Although this procedure is capable of producing a single-celled scatter plot, that plot is best produced using the SG PLOT procedure, which has more features and appearance options.

Unlike the large number statements in the SG PLOT and SG PANEL procedures, the SGSCATTER functionality is contained within three statements:

- The PLOT statement, which creates a panel of independent scatter plots, and has options for fits and computed ellipses
- The COMPARE statement, which creates a shared axis panel of scatter plots, and has options for fits and computed ellipses
- The MATRIX statement, which creates a scatter plot matrix, and has options for computed ellipses and diagonal plots

The SGSCATTER statements define the layout of a single panel, without any paging. However, because the BY statement is supported by the SGSCATTER procedure, you can still create a paging effect using BY variables. The

variable specifications on the COMPARE statement and MATRIX statement define the layout of the panel, while the PLOT statement automatically balances the panel as much as possible to maintain a square layout. You can override this effect of the PLOT statement by specifying either the ROWS or COLUMNS option on the statement.

THE PLOT STATEMENT

The PLOT statement is best used when there is a relationship between the variables that you want to plot, but the data ranges are independent. This statement enables you to put all of the plots in one panel for better examination and comparison.

Figure 15 provides an example. Notice that the group variable is applied to all plots in the panel. A default 95% prediction ellipse is also requested.

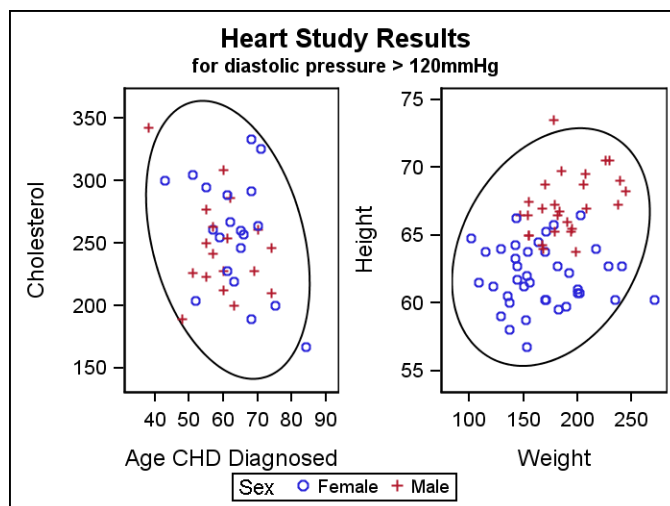


Figure 15: The PLOT statement with a prediction ellipse

```
title1 "Heart Study Results";
title2 h=8pt "for diastolic pressure > 120mmHg";
proc sgscatter data=sashelp.heart (where=(Diastolic>120));
  plot cholesterol*weight Diastolic*AgeCHDdiag / group=sex ellipse;
run;
```

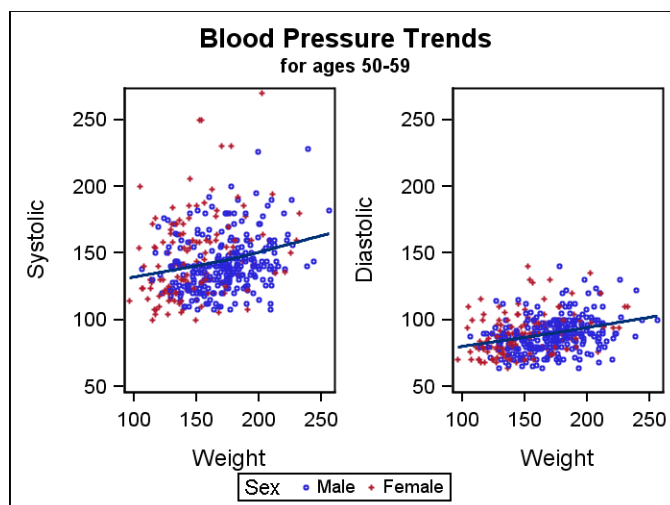


Figure 16: The PLOT statement with the UNISCALE option

You might sometimes want the scale of one or both axes to be uniform across all plots. This setting can be made through the UNISCALE option (see Figure 16).

```

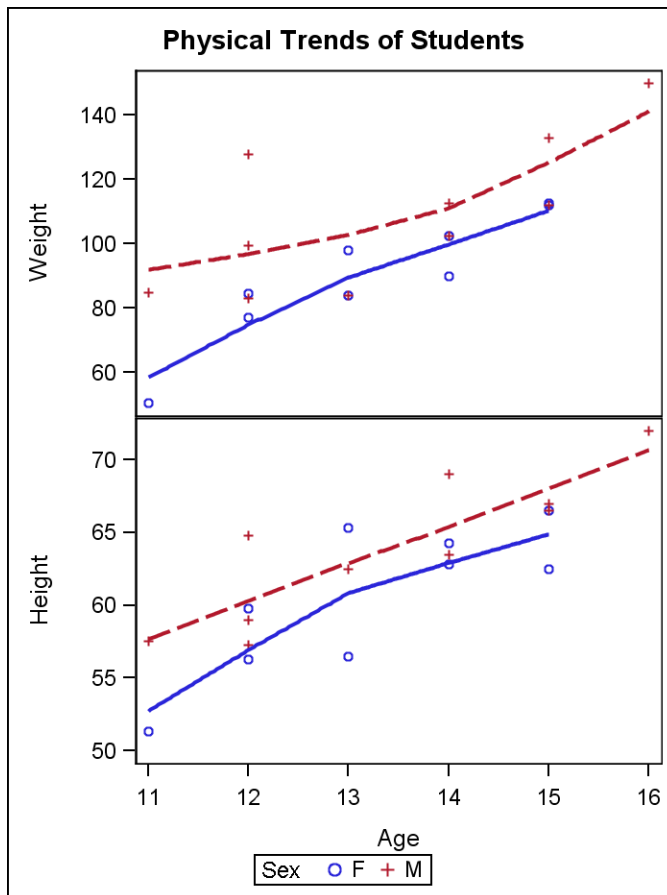
title1 "Blood Pressure Trends";
title2 h=8pt "for ages 50-59";
proc sgscatter data=sashelp.heart;
  where AgeCHDdiag>=50 and AgeCHDdiag<60;
  plot (systolic diastolic)*weight / group=sex loess=(nogroup)
      uniscale=Y markerattrs=(size=3);
run;

```

In Figure 16, we set the blood pressure (Y) axis to be uniform so that the systolic and diastolic pressures maintain their correct relationship against the axes. In addition, we set the LOESS option to ignore the group and fit the entire data.

THE COMPARE STATEMENT

The COMPARE statement is used to create a comparison (MxN) matrix in which the axes are shared across all rows and down all columns. As an example, Figure 17 illustrates how you can compare the height and weight trends between males and females with the ages displayed on a common X axis. Unlike Figure 16, we allow the loess fit to honor the grouping variable when computing the fits.



```

title1 "Physical Trends of Students";
proc sgscatter data=sashelp.class;
  compare y=(weight height) x=age /
    group=sex loess;
run;

```

Figure 17: The COMPARE statement using a shared X axis

THE MATRIX STATEMENT

The MATRIX statement is used to create scatter plot matrices. The list of variables specified on the statement is crossed to create an NxN matrix. This statement also supports computed ellipses and a DIAGONAL option for adding plots in the diagonal.

This type of plot can be useful for finding possible trends or correlations in different plot pairs. In Figure 18, the plot in the petallength/petalwidth crossing suggests that the petal width and petal length might be highly correlated, giving a direction for further analysis.

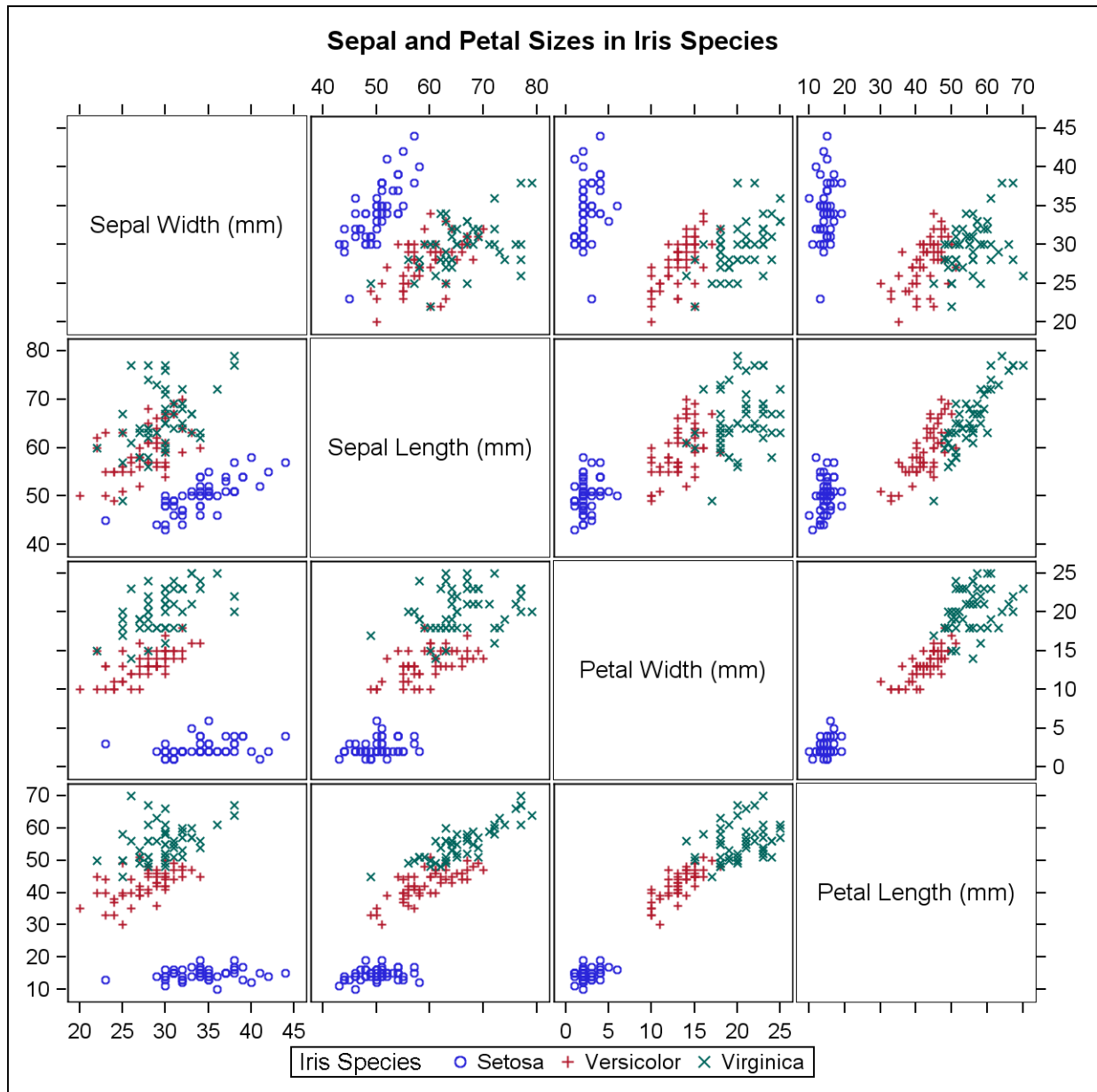


Figure 18: Scatter plot matrix of Iris data

```

title1 "Sepal and Petal Sizes in Iris Species";
proc sgscatter data=sashelp.iris;
  matrix sepalwidth sepallength petalwidth petallength / group=species;
run;

```

As we mentioned earlier, the MATRIX statement enables you to put plots in the diagonal to view the distribution of your matrix variables while viewing the pattern of your scatter points. You can overlay different combinations of histograms, normal curves, and kernel density estimates. In Figure 19, notice again that the petal width and petal length appear to have similar distributions.

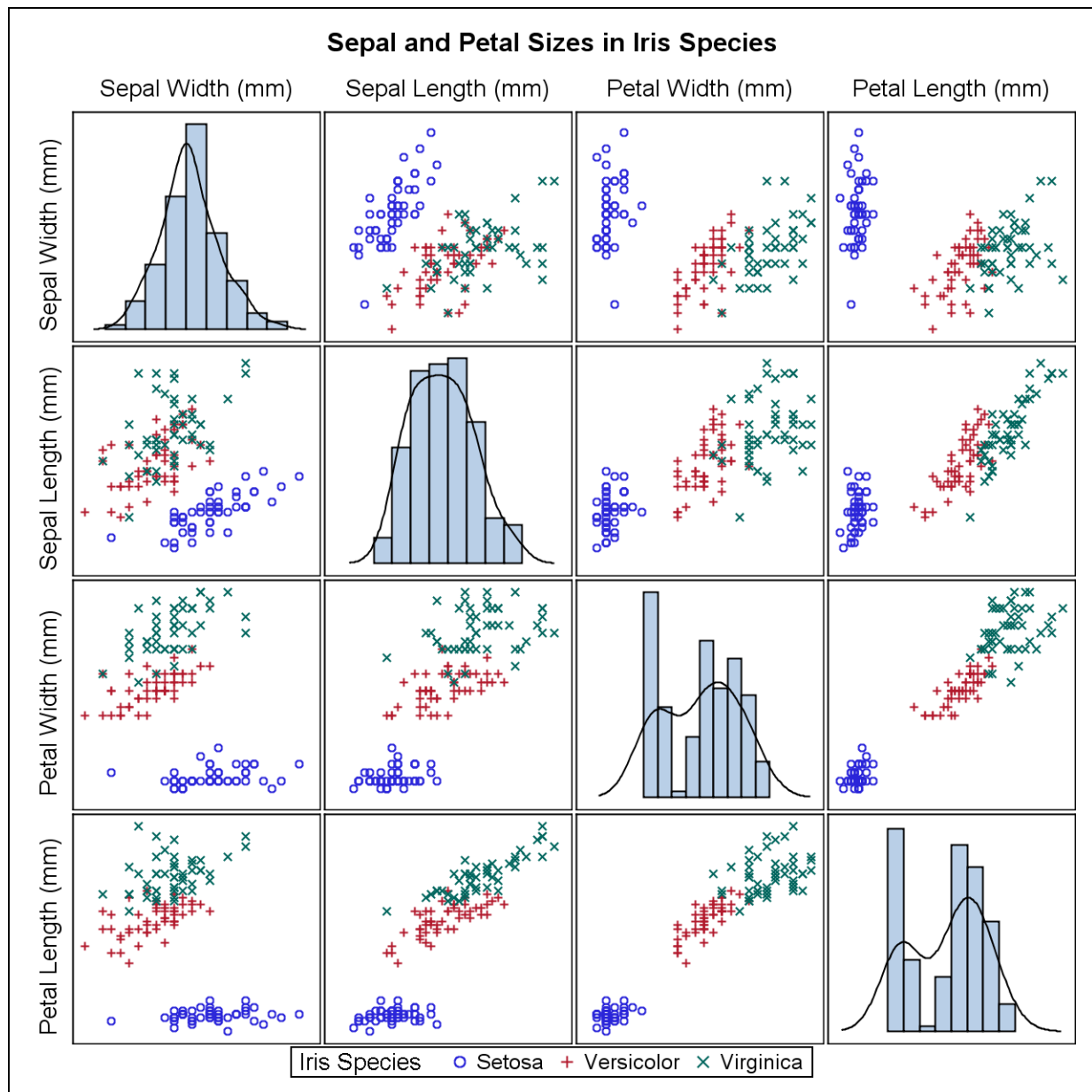


Figure 19: The MATRIX statement using the DIAGONAL option

```

title1 "Sepal and Petal Sizes in Iris Species";
proc sgscatter data=sashelp.iris;
  matrix sepalwidth sepallength petalwidth petallength / group=species
                                         diagonal=(histogram kernel);
run;

```

CONCLUSION

There are many techniques available for making graphs more effective. The techniques that you choose ultimately depend on the audience, the message, and how to best present the information in a way that can be clearly understood. The SG procedures provide a new tool to create effective graphs quickly and with minimal effort, using features such as overlaying, graph layout support and effective style application. I hope you find the techniques described in this paper useful in your own work.

REFERENCES

Robbins, Naomi B. 2005. *Creating More Effective Graphics*. John Wiley & Sons, Inc., New Jersey.
Schwartz, Susan 2008. *Butterflies, Heat Maps, and More: Explore the New Power of SAS/GRAPH*. SAS Global Forum 2008 Proceedings.

ACKNOWLEDGMENTS

I wish to thank Susan Schwartz, Sanjay Matange, and Jeff Cartier for their input and review of this paper.

RECOMMENDED READING

Robbins, Naomi B. 2005. *Creating More Effective Graphics*. John Wiley & Sons, Inc., New Jersey.
Schwartz, Susan 2008. *Butterflies, Heat Maps and More: Explore the New Power of SAS/GRAPH*. SAS Global Forum 2008 Proceedings.
Heath, Dan 2007. *New SAS/GRAPH Procedures for Creating Statistical Graphics in Data Analysis*. SAS Global Forum 2007 Proceedings. <http://support.sas.com/rnd/datavisualization/papers/sgf2007/SGF2007-Proc.pdf>

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Dan Heath
SAS Institute Inc.
500 SAS Campus Drive
Cary, NC 27513
E-mail: Dan.Heath@sas.com

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