Paper HW-04-2013

Using SAS® ODS Graphics

Chuck Kincaid, Experis Business Intelligence and Analytics, Portage, MI

ABSTRACT

This presentation will teach the audience how to use SAS® ODS Graphics. Now part of Base SAS, ODS Graphics are a great way to easily create clear graphics that allow any user to tell their story well. SGPLOT and SGPANEL are two of the procedures that can be used to produce powerful graphics that used to require a lot of work. The core of the procedures are explained, as well as the options available. Furthermore, we explore the ways to combine the individual statements to make more complex graphics that tell the story better.

Any user of Base SAS on any platform will find great value from the SAS ODS Graphics procedures.

INTRODUCTION

SAS/Graph® has been a standard in analytical reporting since the early days. Over the years it has improved in many ways. One of the recent improvements has been the addition of the Graphics Template Language (GTL). Based on the templates used for the Output Delivery System, the GTL gives users amazing control over the look and feel of the graphics that they create. However, the GTL is not necessarily for the faint of heart. To help people use this capability more easily, SAS has created the ODS Graphics procedures: SGPLOT, SGPANEL, SGSCATTER, SGRENDER and SGDESIGN. With thoughtful default values for color and layout, these procedures allow the user to create powerful graphics easily.

The SGPANEL procedure, in particular, introduces the new capability of the *Panel*. The concept of the panel is not new and many SAS Global Forum papers have been written over the years to implement the panel concept, usually using PROC GREPLAY. SGPANEL removes the headaches that have been associated with creating multiple, related graphs. This paper will explain the concepts and capabilities of using SGPANEL and particularly its PANELBY statement. Once the concepts and techniques are mastered, you will never look at graphs the same way again.

SGPLOT

OVERVIEW

The standard output created by ODS Graphics, a.k.a. the SG procedures, are in typical formats, such as PNG and JPEG, which is great for using in other applications and on the web.

The ODS Styles control the overall look and feel of the SG plots giving very nice defaults. The user still has a lot of control within the SG procedures, and can use GTL if they need more.

As we will see, the graph types can be thought of as building blocks. Each graph type can be used on its own and some can be used together to provide more information in a single plot. This concept is similar to the *overlay* from SAS Graph, but better.

UNDERSTANDING PLOT TYPES

There are two kinds of statements within the SGPLOT procedure. The primary statements used to create specific plots and the supporting statements used to enhance or control the plot.

Primary		Supporting
BAND	LOESS	INSET
BUBBLE	NEEDLE	KEYLEGEND
DENSITY	PBSPLINE	LINEPARM
DOT	REG	REFLINE
ELLIPSE	SCATTER	VECTOR

Primary		Supporting
HBAR	SERIES	WATERFALL (preproduction in 9.3)
HBARPARM	STEP	XAXIS
HBOX	VBAR	X2AXIS
HIGHLOW	VBARPARM	YAXIS
HISTOGRAM	VBOX	Y2AXIS
HLINE	VLINE	

As we've said, the primary statements (as this paper calls them) can be combined to make more informative graphs. However, not all statements can be combined with all others. There are four categories of primary statements.

Plot Type	
Basic Plots	SCATTER, SERIES, STEP, BAND, NEEDLE, VECTOR
Fit and Confidence Plots	LOESS, REG, PBSLINE, ELLIPSE
Distribution Plots	xBOX (V or H), HISTOGRAM, DENSITY
Categorization Plots	DOT, xBAR (V or H), xLINE (V or H)

They can be combined with each other in the following combinations

Plot Type	Basic	Fit and Confidence	Distribution	Categorization
Basic Plots	X	X		
Fit and Confidence Plots	X	X		
Distribution Plots			X	
Categorization Plots				X

There are two basic forms for the syntax for these plot types.

First Form – Distribution Plots and Categorization Plots

```
PROC SGPLOT < option(s)>;
<plot keyword> <classification|response variable>
    </ RESPONSE=quantitative variable otheroptions>;
RUN;
```

Second Form – Basic Plots and Fit and Confidence Plots

```
PROC SGPLOT < option(s)>;
<plot keyword> X=<quantitative variable> Y=<quantitative variable> / <options>;
RUN;
```

Finally, before we go into more detail, SAS Help for these procedures can be found in SAS 9.2 under **SAS/GRAPH > SAS/GRAPH 9.2: Statistical Graphics Procedures Guide**, and in SAS 9.3 under **Base SAS > ODS Graphics**. Differences between the two can be found in the SAS 9.3 Help and Documentation under **What's New in SAS 9.3 > Base SAS > ODS Graphics Procedures**.

SGPLOT EXAMPLES

Before we go into the examples, the graphics settings used to create the figures below is the following

```
ods graphics on / reset=all imagename="Figure" width=3in;
```

```
ods html close;
ods listing gpath="&homedir." image_dpi=200 ;
```

The HTML destination is the ODS default in SAS 9.3. However, for this paper, the LISTING destination was all that was needed, so the HTML destination was closed and the LISTING destination opened. The path for the graphic output was set with GPATH and the name of the images was set to "Figure." The first image created will be called "Figure.png." The second and third will be "Figure1.png" and "Figure2.png," respectively, and so on.

The image resolution (DPI or dots per inch) was set to 200 and the width of the image was also set to 3 inches, which gives the easy to read (it is hoped!) images in this paper.

Note that you do not have to turn ODS GRAPHICS ON in SAS 9.3, though you do in SAS 9.2.

As shown above the syntax for these plots is straightforward. To create a dotplot of count of PRODUCT_LINE from the Orion Sales data (SASHELP.ORSALES), we execute the following code, which gives Figure 1.

```
proc sgplot data=sashelp.orsales;
    dot product_line ;
run;
```

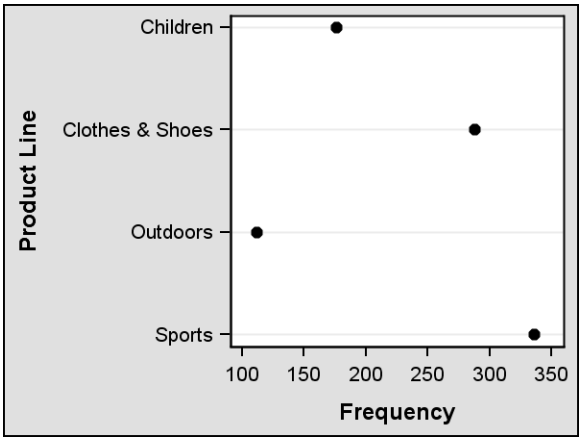


Figure 1 Dot Plot

Examples of code for some of the other plots are below.

Series Plot

```
proc sgplot data=sashelp.stocks (where=(stock = "IBM"));
    series x=date y=close ;
run;
```

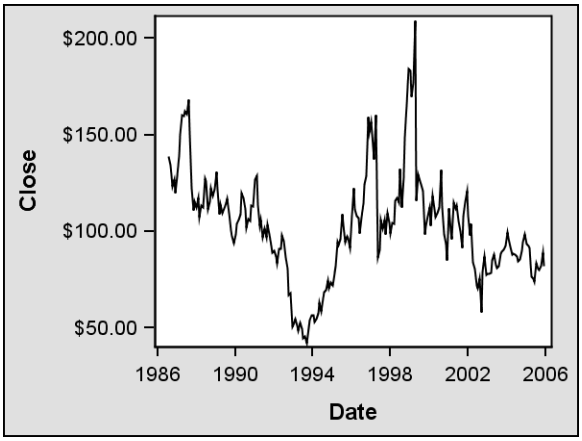


Figure 2 Series Plot

Band Plot

```
proc sgplot data=sashelp.stocks (where=(stock = "IBM"));
    band x=date lower=low upper=high;
run;
```

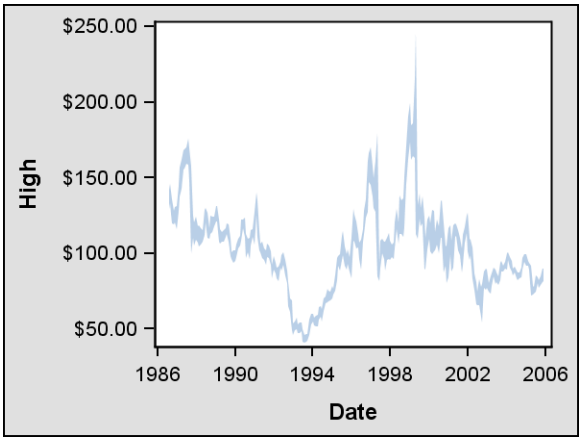


Figure 3 Band Plot

Series Plot

```
proc sgplot data=sashelp.orsales;
    vbox profit;
run;
```

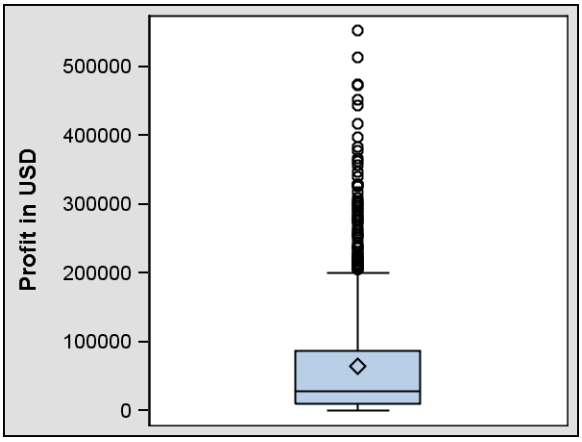


Figure 4 Box Plot

GROUP Option

Each of the plot statements have options that add other information, change the view of the plot, etc. One useful option, across all of the plot statements except HBOX and VBOX, is the GROUP option. This option allows the programmer to plot the same data separately for different groups. For example, take the SERIES plot above. If we do not subset for *stock*, then we have data for IBM, Intel and Microsoft. To see a SERIES plot for all three stocks, the following code is executed. Note that the only differences from above are no subsetting and the GROUP option.

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

This code gives the following plot.

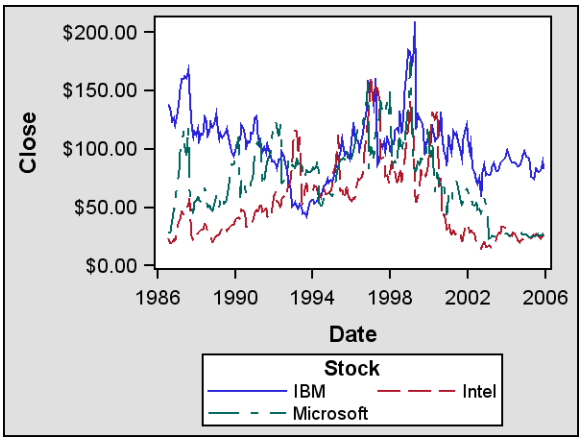


Figure 5 GROUP Option

Notice that SAS selects line colors and styles that make it easy to discriminate among the different stocks. It also supplies a legend by default.

The HBOX and VBOX do their grouping via the CATEGORY option. A separate box plot is created for each level of the variable specified.

One of the many additions in SAS 9.3 is the greater capability with controlling groups. There are 3 new options: CLUSTERWIDTH=, GROUPDISPLAY=, and GROUPORDER= that gives the user more control over the way the groups are laid out when using the GROUP= option and discrete axes. Two examples of these options are shown below for box plots and series plots.

Below we create a series of side-by-side box plots for data broken out by two discrete or categorical variables. Profit is shown for each year (1999 – 2002) and for each Product Line. The default value for GROUPDISPLAY is CLUSTER, which gives us the side-by-side box plots of Product Line for each year.

```
proc sgplot data=sashelp.orsales;
    vbox profit / category=year group=product_line;
run;
```

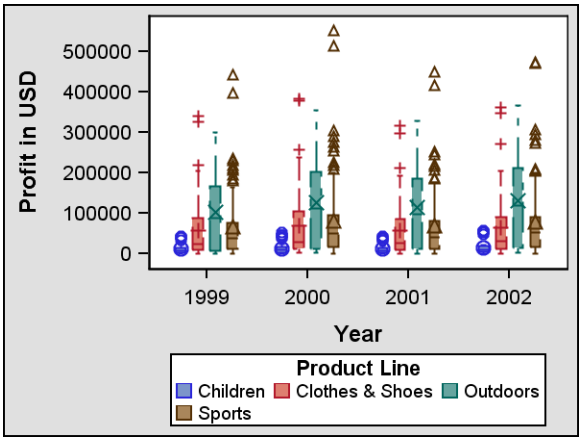


Figure 6 Group and Category for Box Plots

For bar charts the default is different. Below is a similar example.

```
proc sgplot data=sashelp.orsales;
    hbar year / response=quantity group=product_line;
run;
```

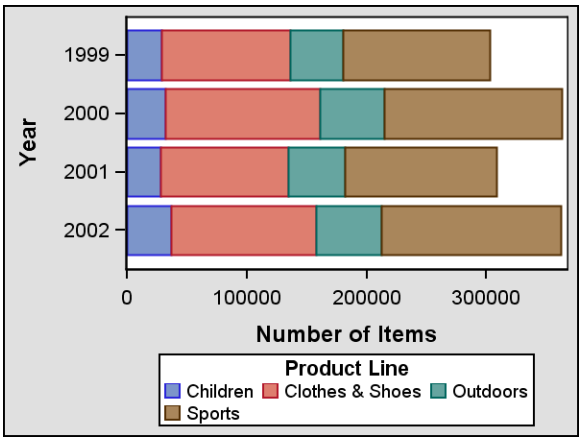


Figure 7 Grouped Bar Chart

As you can see the bars are stacked by default and not side-by-side. Now the GROUPDISPLAY=CLUSTER option is needed to get that effect.

```
proc sgplot data=sashelp.orsales;
    hbar year / response=quantity group=product_line groupdisplay=cluster;
run;
```

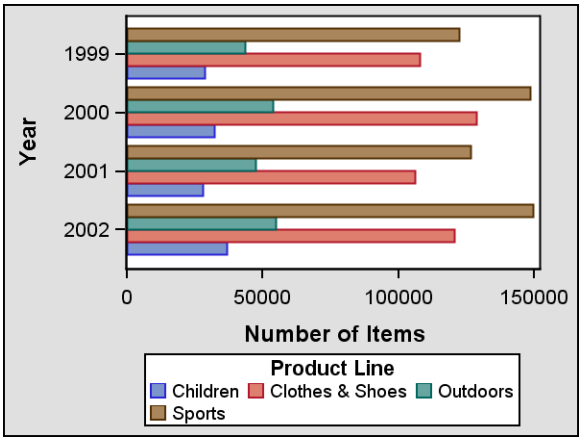


Figure 8 Clusted Groups in Bar Chart

SUPPORTING STATEMENTS

The statements that this paper calls “supporting” statements provide ways to enhance or control the plot. As with the plot statements, there is not enough space in this paper to go through all of the capabilities. We will highlight a few here.

Axes

The axis statements have 20 options that can be used to control its various aspects. Some are similar to ones in the SAS/GRAPH axis statement, but the statements are used differently. With the SG procedures, the XAXIS, YAXIS, X2AXIS, and Y2AXIS statements are in effect only for the current procedure call. The following example creates a log-log scatter plot using the XAXIS and YAXIS statements.

```
proc sgplot data=sashelp.orsales;
    scatter x=quantity y=profit;
    xaxis type=log grid;
    yaxis type=log grid;
run;
```

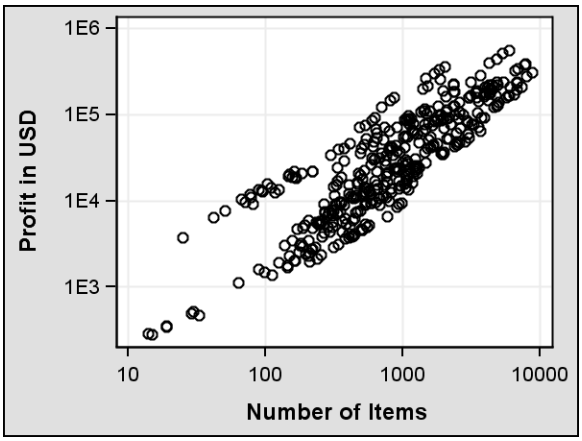


Figure 9 Log type XAXIS and YAXIS

We have also added a grid to the plot, which is barely seen in this medium using the default values. (This problem could be modified with reference lines. That exercise is left to the reader.) Note the display choices used in the plot. The points are not single small points, but small circles. This makes it easier see each point and to tell when there are multiple points at similar locations (until there get to be too many). In addition, the grid is, by default, a lighter color than the points. This allows them to be used as a reference, but not to interfere with our understanding of the data's story.

The programmer can also control the range of the axis (MIN=, MAX=), the values of the tickmarks (VALUES=), and how they are displayed when there are too many (FITPOLICY). A nice option for time series data is INTERVAL=. This option determines the tick interval for the time axis. SECOND, TENDAY, and QUARTER are three examples.

KEYLEGEND

The KEYLEGEND statement allows the programmer to control what is displayed in the legend, where it is displayed, and how it is displayed. If no legend at all were desired, then the programmer would use the PROC SGPLOT option NOAUTOLEGEND.

The six KEYLEGEND options are straightforward. The optional arguments are new and interesting. The plot statements can be assigned a name so that they can be referenced in the KEYLEGEND statement. This allows the programmer to associate only certain parts of the plot with a legend or show, for example, two legends for two different parts of the plot. Below is an example of the former (with other options included for your edification).

```
proc sgplot data=sbs_stocks;
    scatter x=ibm_vol y=ibm_high / name="ibm_s" legendlabel="IBM";
    scatter x=intel_vol y=intel_high / name="intel_s" legendlabel="Intel";
    scatter x=msft_vol y=msft_high / name="msft_s" legendlabel="Microsoft";
    ellipse x=ibm_vol y=ibm_high / name="ibm_e" clip;
    ellipse x=intel_vol y=intel_high / name="intel_e" clip;
    ellipse x=msft_vol y=msft_high / name="msft_e" clip;
    keylegend "ibm_s" "intel_s" "msft_s" / position=bottomright location=inside;
    inset "With 95% Prediction Ellipses";
    xaxis label="Daily Volume";
    yaxis label="Daily High" grid refticks;
run;
```

Notice that there are three SCATTER statements and three ELLIPSE statements, each with the NAME= option. Without the KEYLEGEND statement, all six would be identified in the legend. Since we identify the names of the plots that we want included with the legend – "ibm_s", "intel_s" and "msft_s" – the legend only shows those symbols. See Figure 10 below.

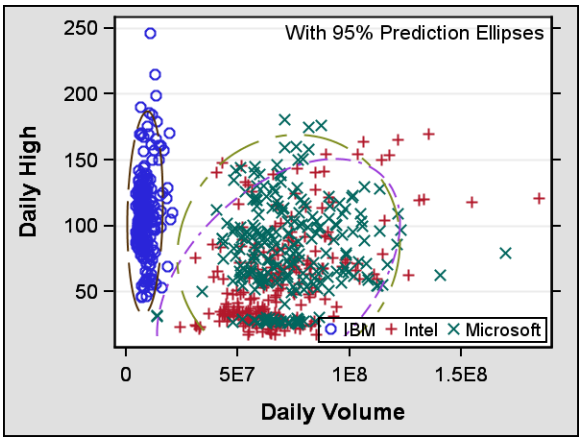


Figure 10 Legend Control with KEYLEGEND

Notice the colors and symbols used for the different plots. The SAS logic evaluates the number and kind of plots, and selects colors and symbols that provide high readability. The programmer does have control over these options. However, most of the time the defaults will be fine.

SGPLOT has other options and capabilities that the programmer will likely want to use. Other SAS Global Forum papers and the SAS Help Documentation are very good sources for information.

COPLOTS

WHAT IS A COPLOT?

The Coplot is a graphical device that has its origins in E. R. Tufte's *small multiples*. Tufte discusses this graphic technique in his book The Visual Display of Quantitative Information (1983). He compares them to frames of a movie: "a series of graphics, showing the same combination of variables, indexed by changes in another variable." By keeping the design of the graphic constant throughout all the frames, the viewer can focus on the changes in the data as they visually move from one frame to another.

Figure 11 below shows an example of small multiples that compares the distribution of *Age of Herring Catches* each year from 1908 to 1913. The shift of the peak (representing the tremendous number of herring that were spawned in 1904) throughout the years can be easily seen, showing the power of this type of plot.

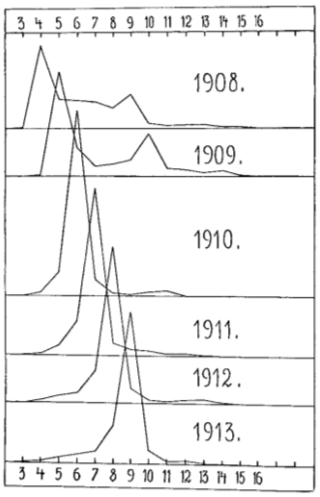


Figure 11 Tufte's small multiples example

William S. Cleveland brought his ideas together with other research and introduced the concept of the *conditioning plot* or *coplot* in Visualizing Data (1993). Figure 12 shows the first coplot Cleveland introduces. This figure has the characteristic *given panel* across the top of the graph, and the *dependence panels* in the center.

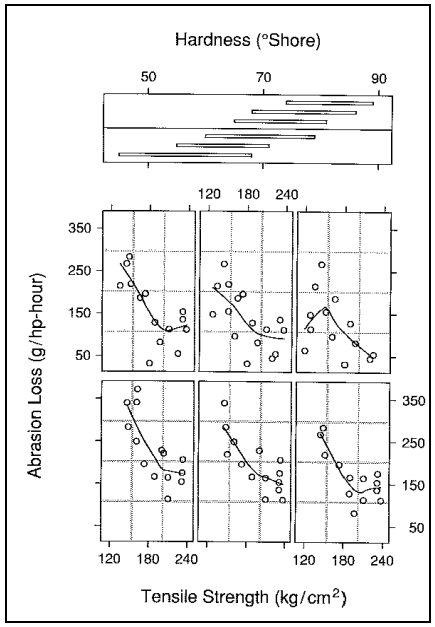


Figure 12 Cleveland's Rubber Example

Each dependence panel graphs, in this case, two variables, *Abrasion Loss* vs *Tensile Strength* for a specific range of values for *Hardness*. One can easily see the conditional dependence of the relationship between *Abrasion Loss* and *Tensile Strength* upon *Hardness*. Most of the panels show a particular nonlinear "hockey-stick" relationship, except for the top-right panel, which shows an inversion of the relationship.

Different software applications each have a different way of presenting this powerful graphing technique. In this paper, we will look at what SAS does in SGPANEL and what can be done with it.

SGPANEL VS SGPLOT

SGPLOT is available for creating the typical one panel plot. All of the plot types that SGPLOT can create, except the ELLIPSE plot, are available in SGPANEL.

The primary difference between SGPLOT and SGPANEL is that the former gives one plot in one panel, while the latter creates multiple plots laid out in multiple panels. The way the layout is controlled with the PANELBY statement, which we will go into below. Other documented differences include

- **No TMPLOUT="filename" option for SGPANEL.** This option writes the GTL code to a file that can be modified and used later on. However, even though it is not documented, it does work.
- **No UNIFORM= option.** This option specifies how to control axis and markers with the BY statement. The BY statement can be used with SGPANEL and there are some times where it is better to do so. However, the UNIFORM option is not accepted. Because of the *panel* concept, SGPANEL makes the group values and axis scaling consistent by default. However, the scaling can be controlled as described below.
- **KEYLEGEND.** The user has less control over the legend, with KEYLEGEND, in SGPANEL. The legend in SGPLOT can be placed either inside or outside the graph. For SGPANEL the legend is relevant to all panels, so it only makes sense to have it "outside" all of the panels, but within the plot region. Further, the possible positions for the legend in SGPANEL are BOTTOM, LEFT, RIGHT, and TOP, whereas SGPLOT can use some combinations.

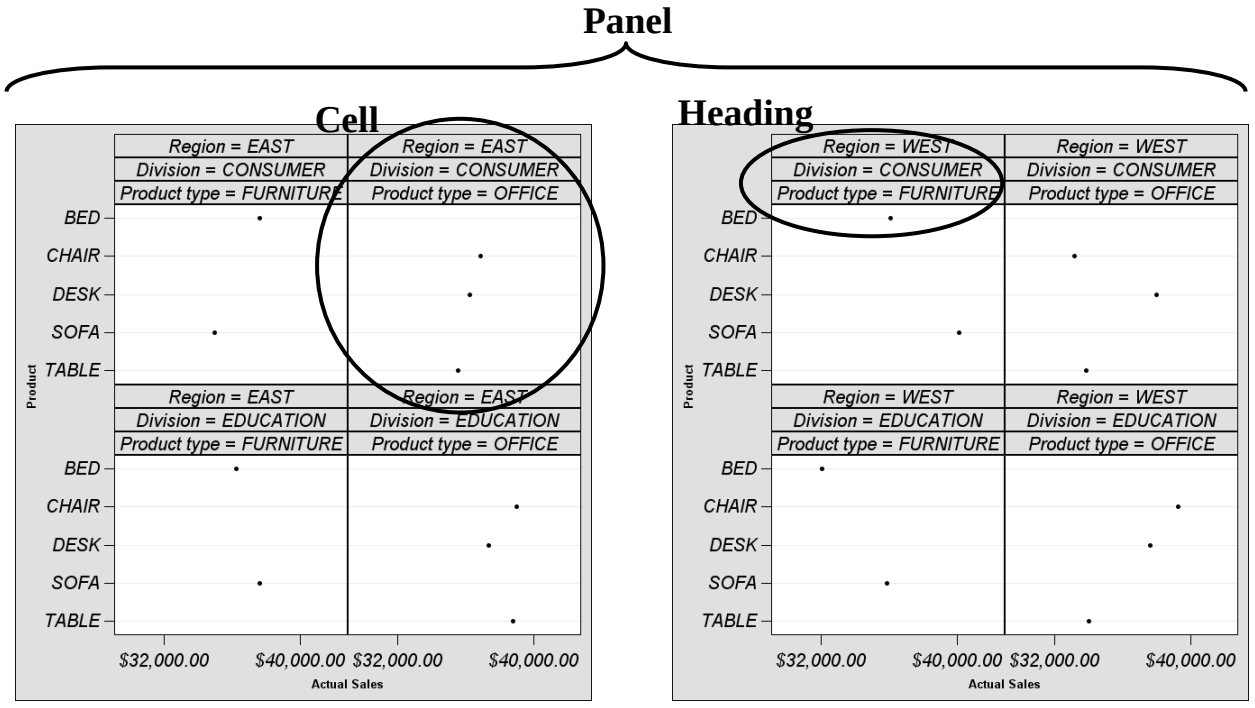
SGPANEL AND PANELBY

SGPANEL displays the relationship among specified variables (cf. dependence panels, above) at given *levels* of another variable (cf. given panel, from above). The variable being conditioned is expected to be a class variable

where the *levels* are its unique values. Numeric variables can be used, but the procedure will attempt to make a chart for each unique value, so be careful!

First, let's review the terminology of the SG Procedures. There are potentially three levels of SG output. (Not to be confused with levels of a class variable.) The most basic is the *cell*. The *cell* is what has been traditionally called a graph such as created with PLOT, GPLOT or even SGPLOT. The *panel* is the set of all cells that the procedure generates. Finally, the *graph* is one page of output. For example, when a panel is split across multiple pages, each page is a *graph*.

The labels at the top, in this case, of each cell are the *headings* or *header*. Figure 13 provides visual examples for each of these terms.



Page 1 == Graph 1

Page 2 == Graph 2

Figure 13 Terminology Example

THE PANELBY STATEMENT

The PANELBY statement is required and must be the first statement before any plot, axis or legend statements. The procedure will use what is specified in the PANELBY statement to define the overall layout of the panel. The syntax for PANELBY is

PANELBY variable(s) </ option(s)>;

option(s) can be one or more of the following:

BORDER | NOBORDER

COLHEADERPOS= TOP | BOTTOM | BOTH

COLUMNS= n

LAYOUT= LATTICE | PANEL | ROWLATTICE | COLUMNLATTICE

MISSING

NOVARNAME

ONEPANEL

ROWHEADERPOS= RIGHT | LEFT | BOTH

ROWS= n

SPACING= n

SPARSE

START= TOPLEFT | BOTTOMLEFT

UNISCALE= ROW | ALL

In this paper, we will cover many of these options and explore what benefit they provide the user.

LAYOUT= LATTICE | PANEL | ROWLATTICE | COLUMNLATTICE

One of the more important options is the LAYOUT. The two primary forms of the panel layout are LATTICE and PANEL (the default). Here is an example of the PANEL layout using the Orion Sales data in SASHELP.

```
proc sgpanel data=sashelp.orsales;
    panelby product_line;
    dot year / response=profit;
run;
```

This default layout arranges the cells alphabetically according to the classification variable. The order is by row from left to right and top to bottom with the headings placed at the top of each cell. The example panel is shown in Figure 14. This layout can be controlled with the START= option below. The number of columns per row is determined by the number of levels of the classification variable. Right now the procedure uses each level of the classification variable to create a cell. Eventually, it may take numeric variables and create intervals that it could use for the classification. Cleveland called this *slicing*. Another paper by the author and Jack Fuller shows how to do slicing on your own. You can create a macro that has the details worked out.

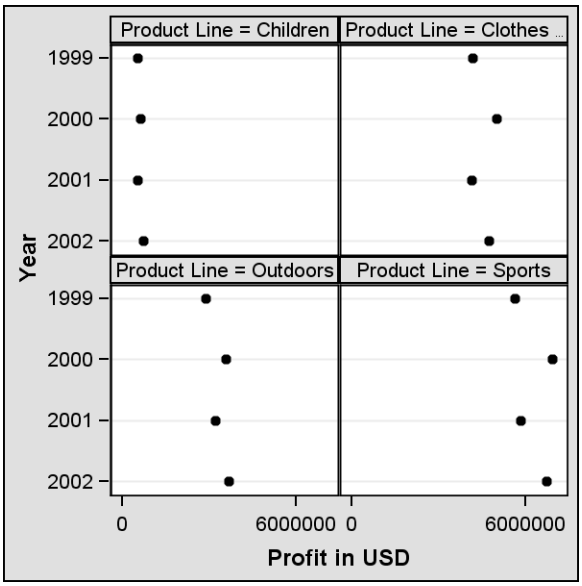


Figure 14 Default Panel Layout

The PROC tries to make intelligent decisions about how to divide the panels into rows and columns. Table 1 shows the default numbers of rows, columns and pages given the number of panels. As can be seen, unless the number of rows or columns is overridden, the procedure keeps them to 3 and under.

Table 1 Default Layout for PANEL Option

# of Panels	# of Rows	# of Columns	# of Pages	# of Panels	# of Rows	# of Columns	# of Pages
2	1	2	1	8	2	2	2
3	2	2	1	9	1	2	5
4	2	2	1	10	1	2	5
5	2	3	1	11	2	3	2
6	2	3	1	12	2	2	3
7	1	2	4				

The layout can be controlled by the options, COLUMNS, ONEPANEL, and ROWS, as we will see. The panel layout is useful for single PANELBY variables. The PANELBY could be used with two variables and controlling the number of columns or rows, however it is easier to do this with the LATTICE layout. The LATTICE layout uses the first variable to define the columns and the second to define the rows. Thus, if the first variable has 2 levels and the second has 3 levels, then SGPANEL will create a layout with 2 columns and 3 rows.

By default, the following number of cells gives the corresponding numbers of rows, columns and pages.

Table 2 Default Layout for LATTICE Option

# of Levels 1st Var	# of Levels 2nd Var	# of Rows	# of Columns	# of Pages	# of Levels 1st Var	# of Levels 2nd Var	# of Rows	# of Columns	# of Pages
2	2	2	2	1	4	2	2	2	2
2	3	2	3	1	4	3	3	2	2
2	4	2	2	2	4	4	1	2	8
2	5	1	2	5	4	5	1	2	10
3	2	2	3	1	5	2	2	1	5
3	3	1	3	3	5	3	3	1	5
3	4	2	3	2	5	4	2	1	10
3	5	1	3	5	5	5	2	3	6

Note that certain layouts can leave some cells blank depending on the number of levels of the given variables. For example, dividing five columns into two pages will mean three columns on each page. (It tries to keep things uniform). This arrangement will then give one blank column on the second page.

As mentioned, the layout can be controlled by using the COLUMNS, ROWS or ONEPANEL option in the following manner.

COLUMNS= N

ROWS= N

These options control the number of columns and/or rows in the layout. The programmer does not have complete control with these options. The procedure will still make its own decisions about the layout if the settings do not fit well into the page. In addition, these options have no effect if the ONEPANEL option and the LATTICE layout is used. Only one of them will have an effect with the ONEPANEL and the PANELBY layout. In that case, if both COLUMNS= and ROWS= are specified, then the COLUMNS= value will be used and the ROWS= value ignored.

ONEPANEL

The ONEPANEL option on the PANELBY statement makes the entire panel fit into a single graph, which is a single page. If the panel has many cells, the graph is going to be cluttered, so this option is typically recommended when there are only a small number of cells. However, many cells can be okay if the pattern of the relationship in each cell and across cells is what is important and not the actual relationship. An example of this from the web is by Dr. Andrew Gelman, Professor of Statistics and Political Science and Director of the Applied Statistics Center at Columbia University.

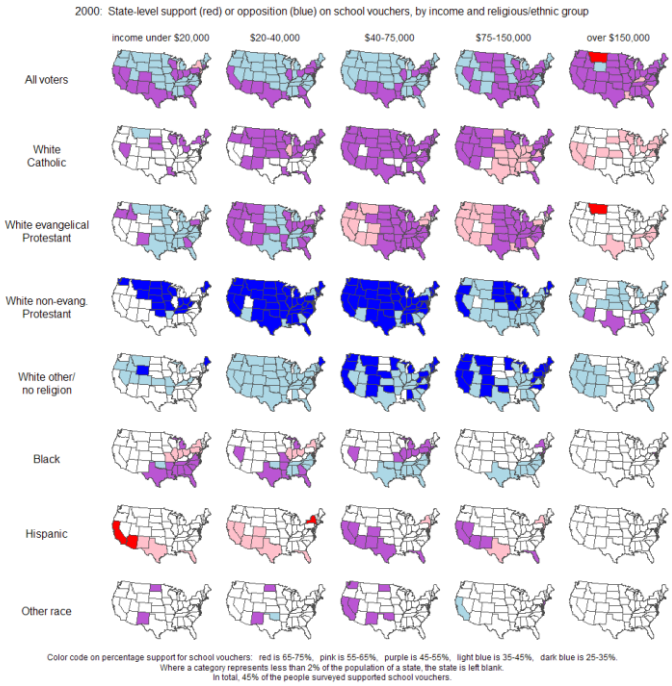


Figure 15 Small Multiples Example 1 (Gelman)

Notice that it is not the individual graphic that we're interested in as much as how the graphic changes as the given variables along the top and side change.

An example of this same concept with SGPANEL can be seen in Figure 16. We could even remove the tick labels here, since they are not of much interest. By using the LOESS statement with NOMARKERS, we can see the overall relationship between the Systolic and Diastolic variables for each of the levels of *Cause of Death* and *Smoking Status*. Most seem similar to each other, but there are a few that stand out.

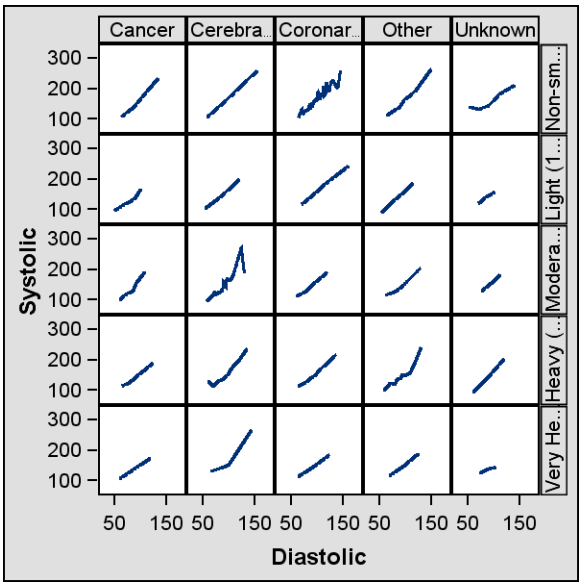


Figure 16 SGPANEL Small Multiples

When used with discretion, this option can be a valuable tool either with many or few cells in your graph. Note, however, that if the panel becomes too large for the output image, then a blank image is created. Finally, with this option and the PANEL layout, both the ROWS= and COLUMNS= values cannot be specified. If they are, then

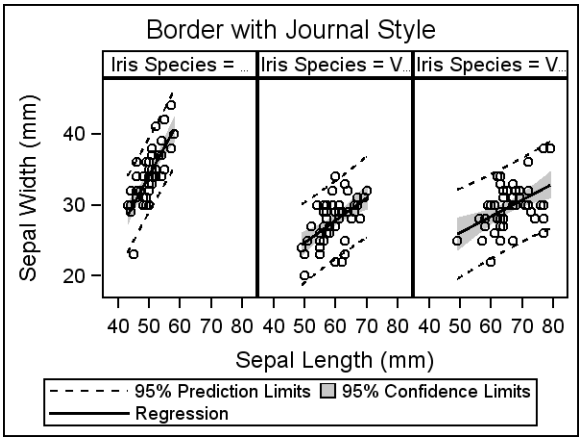
COLUMNS= will be used and ROWS= will be ignored. Neither of these, ROWS= or COLUMNS=, have an effect with ONEPANEL and the LATTICE layout.

BORDER | NOBORDER

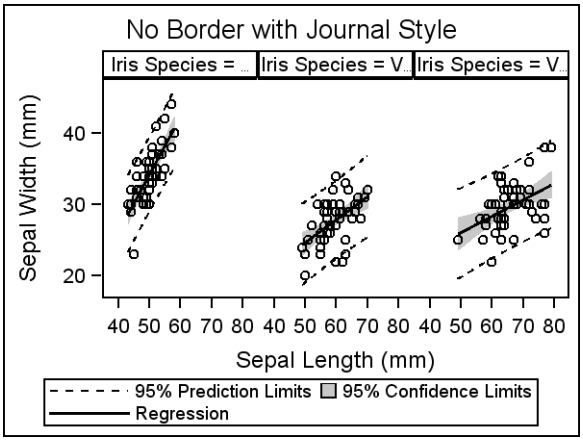
This option is straightforward depending on the ODS style in use at the time. The typical styles (Analysis, Statistical, Listing, Journal, Journal2) all have borders by default. BORDER adds borders around each cell in the panel and NOBORDER removes them. Note that this is around each cell and not around the graph as a whole. Here is an example of each with the *Journal* style.

Table 3 BORDER vs NOBORDER

Border with Journal Style



No Border with Journal Style



COLHEADERPOS= TOP | BOTTOM | BOTH

ROWHEADERPOS= RIGHT | LEFT | BOTH

These options, which (obviously) control the position of the column and row headings, can be used with the LATTICE layout. For the PANEL layout, the headings are always across the top. The default positions (TOP and RIGHT) are the most useful, because the axes can "interfere" with the BOTTOM and LEFT positions. BOTTOM and LEFT can be useful with small multiples if the axes are not displayed as we can see with the small multiples heart data example above. (See Figure 17.)

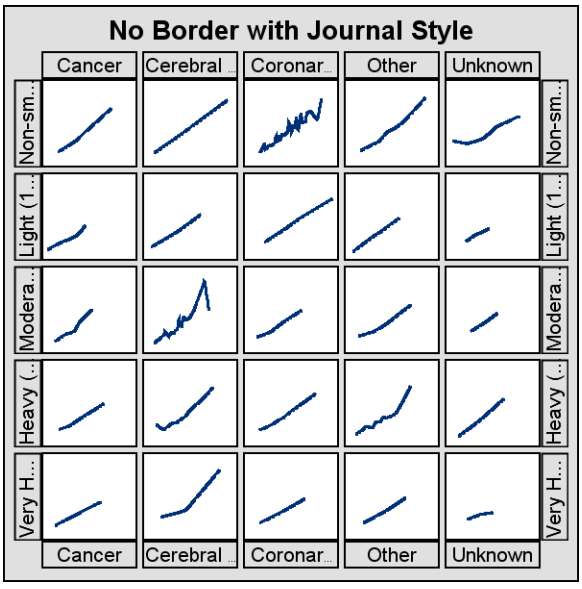


Figure 17 Cell Headers on BOTH Sides

MISSING

Most procedures by default cannot use records with missing values. PROC REG, for example, will drop observations that have missing values for any of the variables in the model. However, sometimes it's helpful to explore the behavior of the non-missing variables for these records. The MISSING option can help by treating the missing values of the PANELBY classification variable as a valid level and a cell will be drawn.

NOVARNAME

The NOVARNAME option is used in almost every graph the author creates, because it removes the variable name and the "=" symbol from the cell headings. This style provides clean heading labels by avoiding the redundant information. If there are two PANELBY variables that have the same levels, however, then the variable name may be necessary.

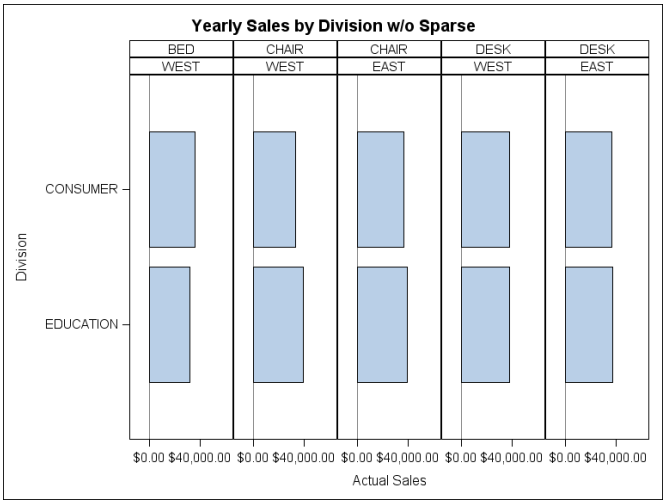
SPACING= N

This option controls the number of pixels between the cells. The author has not had reason to use this option.

SPARSE & MISSING

The SPARSE option is useful when not all combinations of the PANELBY classification variables are there. We return to the Orion Sales Data to explain. We will create a horizontal bar chart with two classification variables. PRODUCT is the first classification variable, so it is on top and "moves" the slowest. REGION is the second classification variable, so it is under PRODUCT and "moves" faster. That is, SGPANEL cycles through the levels of REGION within each level of PRODUCT; very similar to the behavior of PROC SORT. Figure 18 shows our sales example with the data for the BED & EAST and SOFA & EAST combinations deleted (PRDSALE_SUB). Without the SPARSE option the levels are all adjacent. Note that it leaves blank cells at the end to balance the graphs. (These graphs were not done with a width of 3 in.)

```
proc sgpanel data=prdsale_sub;
  title "Yearly Sales by Division w/o Sparse";
  panelby product region / novarname columns=5;
  hbar Division / response=actual;
run;
```



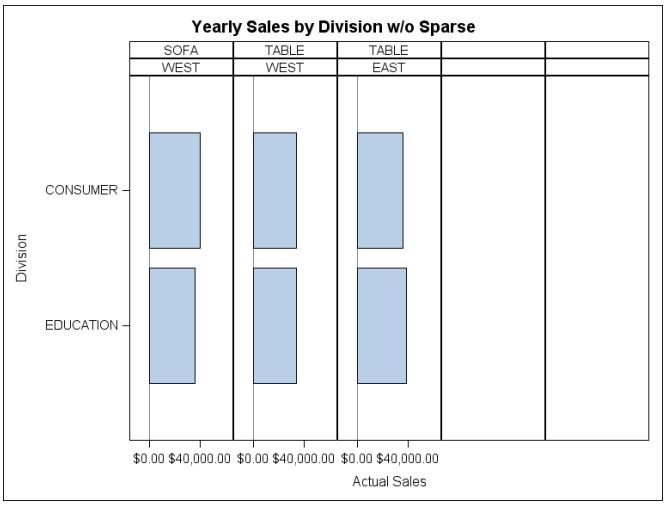


Figure 18 Missing Levels without SPARSE

It is interesting to note the order of the cells in Figure 18. Remember that the order of the levels within a classification variable is alphabetical. The order of the levels within the second classification variable is set by their behavior in the first level of the first classification variable. Because the BED x EAST combination is missing, WEST is first within BED. That makes WEST first within all the other levels of PRODUCT.

When we add the SPARSE option,

```
proc sgpanel data=prdsale_sub;
  title "Yearly Sales by Division w/ Sparse";
  panelby product region / sparse novarname columns=5;
  hbar division / response=actual;
run;
```

the missing combinations are included in the grid as shown in Figure 19.

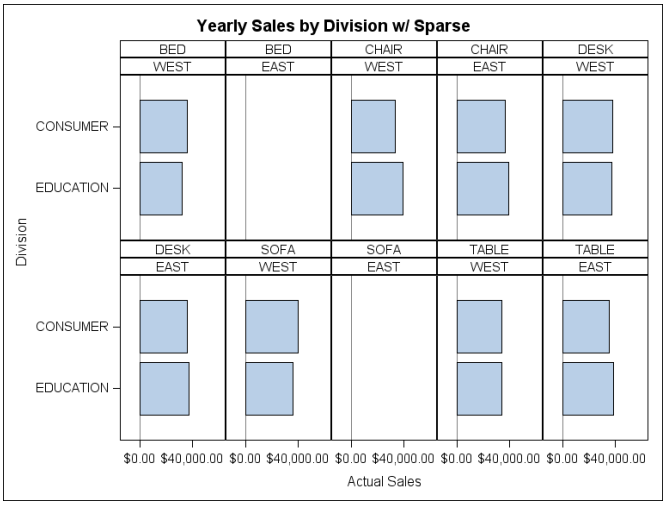


Figure 19 Missing Levels with SPARSE

Note that the order of the cells is essentially the same as before. The cell for the missing combination BED x EAST is displayed after all the other levels for REGION. (In this case there is only WEST.)

In Figure 19 the order is WEST then EAST because the BED x EAST combination is missing. Since that occurred under BED (the first level of PRODUCT) all of the other PRODUCTS are WEST then EAST. If instead, the CHAIR x

EAST combination had been missing (or any of the PRODUCTS other than the first), then all of the PRODUCTS would have had EAST then WEST.

This layout is sort of similar to the LATTICE layout which shows all combinations by default. See Figure 20.

```
proc sgpanel data=prdsale_sub;
  title "Yearly Sales by Division w/ Lattice";
  panelby product region / layout=lattice novarname columns=5;
  hbar Division / response=actual;
run;
```

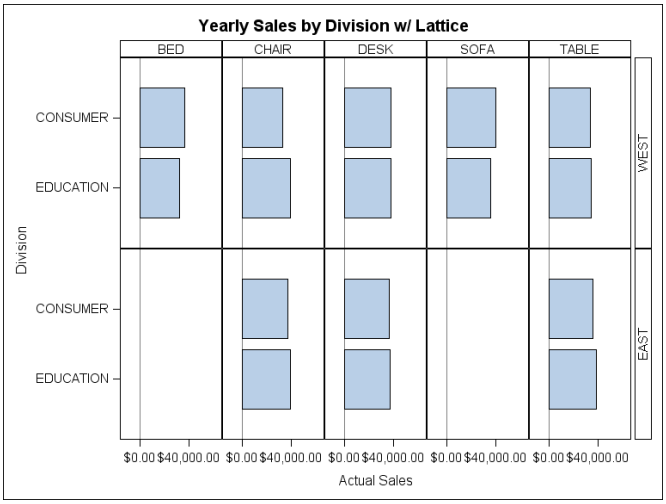


Figure 20 Missing Levels with Lattice Layout

The MISSING option and the SPARSE option sound similar to one another and they are. The difference is in what's missing. Suppose we have a data set with classification variables *A* and *B* that we'll use for defining our cells which will plot *X* vs *Y*. The SPARSE option is useful when no records exist (i.e. no data for *X* and *Y*) for some combinations of *A* and *B*. The MISSING option would be useful when we have data records with values for *X* and *Y*, but the values for *A* and/or *B* are missing.

START= TOPLEFT | BOTTOMLEFT

This option determines the order of the cells in the graphs by specifying the location of the first cell. TOPLEFT (default) is how we read tables and BOTTOMLEFT is graphical order. From the SAS Documentation we have these diagrams.

1	2	3
4	5	6
7	8	9

7	8	9
4	5	6
1	2	3

UNISCALE= COLUMN | ROW | ALL

The default value for this option is ALL and most of the time that's the appropriate one. Specifying either COLUMN or ROW scales just the column axes or row axes, respectively, to be identical. These may be helpful if the scales for the levels of the other dimension are radically different.

COLAXIS AND ROWAXIS

Most of the axis options behave the same with PROC SGPANEL as they do with SGPLOT. We mentioned the DISPLAY=NONE option above when discussing the ROWHEADERPOS and COLHEADERPOS options.

The ALTERNATE option will "interfere" with the row and column headings when using the LATTICE layout. The PANEL layout can benefit from this option, though, particularly if there are a large number of columns or rows. This option will add reference ticks to each side (Top and Bottom, or Left and Right) of the panel and alternates the tick values between the two sides, depending on whether the COLAXIS or ROWAXIS is used.

CONCLUSION

The SGPANEL procedure is a very powerful part of the Statistical Graphics family. It allows the intuitive display of complex, high-dimensional information. Knowing how to use the many options will help you customize your graphs to best present your data. I hope this paper will help you in doing so.

REFERENCES

- Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.
- William S. Cleveland. *Visualizing Data*. Hobart Press. 1993.
- Lora D. Delwiche, Slaughter, Susan J. *Using PROC SGPLOT for Quick High Quality Graphs*. WUSS 2008. http://www.wuss.org/proceedings08/08WUSS%20Proceedings/papers/how/how05.pdf
- Andrew Gelman. *Who wants school vouchers?* June 8, 2009 http://www.stat.columbia.edu/~cook/movabletype/archives/2009/06/estimated_suppo.html

ACKNOWLEDGMENTS

Thank you to the SAS ODS Development team for their ideas in the conception of this paper and for answering questions along the way. Thank you, also, to Jack Fuller for his careful review and suggestions.

CONTACT INFORMATION

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

Chuck Kincaid
Experis Business Intelligence and Analytics Practice
5220 Lovers Lane
Portage, MI 49002
269-553-5140
chuck.kincaid@experis.com
www.experis.us/analytics