Paper 277-2011

Now You Can Annotate Your Statistical Graphics Procedure Graphs

Dan Heath, SAS Institute Inc., Cary, NC

ABSTRACT

Have you ever had a situation where you wanted to add custom labeling or other graphical features to your statistical graphics (SG) procedure output but could not find a way to do it? Even with all of the plot types and options of the SG procedures, you might have run into this situation. With SAS® 9.3, you can add these features directly by using the new annotation support in the SG procedures.

The annotation data set definition used by these procedures has been redesigned to take advantage of ODS Graphics functionality. This paper will discuss this functionality in detail, highlighting new features in this system that include the following:

- rich text support (including superscripts, subscripts, and Unicode)
- transparency support for all primitives
- more drawing spaces for easier placement of annotations

INTRODUCTION

One of the goals of the SG procedures is to surface the power of the Graph Template Language (GTL) in a concise procedure syntax. The SGPLOT and SGPANEL procedures now have over 20 plot types that can be combined in a variety of ways to make creative and complex plots. Some of these plots can even be used to create annotation effects (see HEATH 2009); however, there might be times when these plotting techniques are not sufficient for your needs.

With SAS 9.3, the SGPLOT, SGPANEL, and SGSCATTER procedures support a pre-production annotation facility using annotation data sets. This facility leverages the new DRAW statement functionality of GTL and enables you to include annotations anywhere within the graph. The annotation data set definition has been redesigned to take advantage of GTL functionality, as well as make the column names and values easier to use and remember.

The foundation of an annotate data set is the FUNCTION column. This column specifies which drawing operation to perform. The remaining columns are required or optional based on the function requested, and not all columns apply to all functions. There are currently 45 reserved column names to support the features of all functions; however, you will typically use a small percentage of those columns for any annotation you create. It is also typical to have a number of missing values in your annotate data set when you use multiple function types in the same data set. Not all of the features represented by these columns will be covered in this paper; however, the key features of each function will be covered along with examples of how these functions can be used.

One other note about data set construction: be sure to check the length of your columns. This annotation definition uses a number of varying-length keywords. You might start with an attribute that is only four characters wide and change it to something eight characters wide later in the DATA step. The eight-character value will be truncated to four characters unless you use the LENGTH statement or set a column range in the INPUT statement. If you see odd results in your graph, be sure to check the log to see if invalid attributes are specified due to truncation.

DRAWSPACES

Before creating annotations, it is important to understand the concept of DRAWSPACE. A graph contains multiple drawing areas that are used as the basis for placing the annotation. Figure 1 shows all of the available drawing spaces in an SGPLOT output. The SGPANEL and SGSCATTER procedures support only the graph and layout spaces.

The DRAWSPACE keywords used in the data set are a combination of the drawing area and the drawing unit. All drawing areas support pixel and percentage units. The data drawing area also supports data value units for positioning annotations using axis values. For example, the keyword for drawing an annotation in the wall area using percentage units is WALLPERCENT. These keywords can be specified in the DRAWSPACE column to apply to all coordinates of the drawing function, or they can be specified on specific drawing space columns for each coordinate of the function.

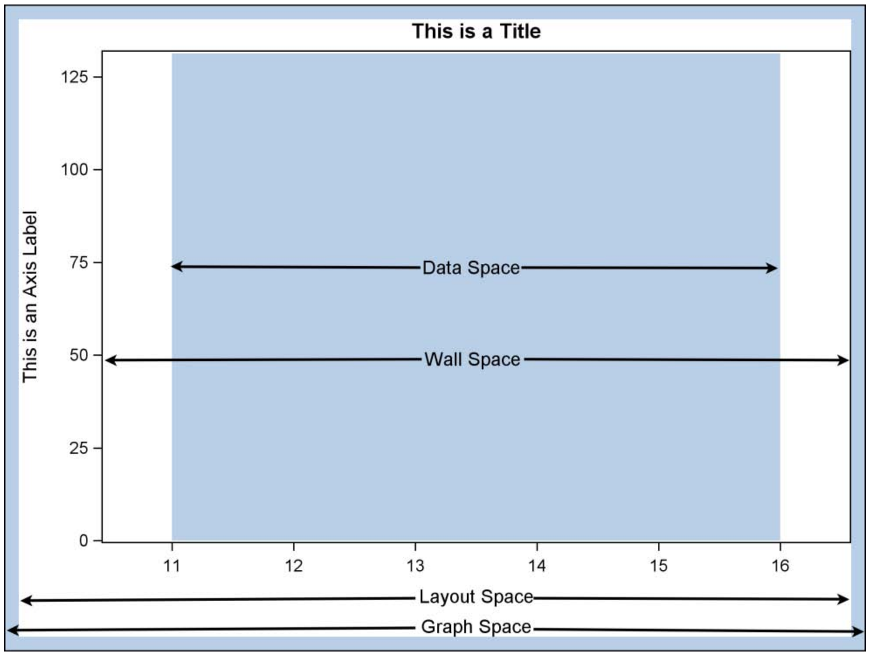


Figure 1. Drawing Spaces in SGPLOT

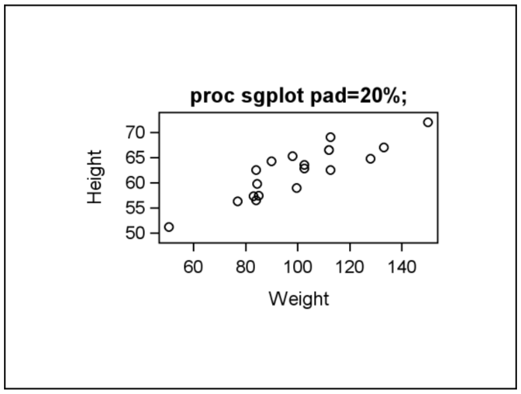
When using percent or pixel units, the origin for all spaces is the bottom left corner. The origin for the DATAVALUE space depends on the direction of the axes. The facility allows you to specify values outside of the normal range to place annotations. For example, specifying WALLPERCENT and an X coordinate of 105% will put the annotation 5% outside of the right edge of the wall. This technique can be useful when you want to bind the placement of an exterior annotation to the position of the wall.

There are several considerations when deciding which drawing space to use for your annotation. These considerations include:

- Should this annotation be bound to a data location along the axis? If so, use the DATAVALUE drawing space for that coordinate.
- Should this annotation be bound to a certain area? If so, use that area's drawing space or the drawing space of an area within that certain area. For example, if you want your annotation to stay within the wall area, but you use the GRAPHPERCENT drawing space, it is possible that a new data set used in SGPLOT could produce longer tick values and push the wall away from your annotation. It's better in that case to use WALLPERCENT.
- Should this annotation always be positioned relative to, but outside of, a certain area? If so, you should use the drawing space of that certain area but use coordinates that extend beyond that area. For example, if you want to put a text value just outside of the wall to the right, and always have it maintain that relative position, you could specify a drawing space of WALLPERCENT and set the X coordinate to 101%.
- Should the same drawing space be used by all of the coordinates of the annotation? If so, use the DRAWSPACE column as a shortcut to specify your drawing space for the annotation.
- Does the annotation need to draw in more than one space or use different units for each coordinate? If so, you will want to specify the drawing space in each coordinate's drawing space column. These columns are X1SPACE, Y1SPACE, X2SPACE, and Y2SPACE. The X2SPACE and Y2SPACE columns apply only for the LINE and ARROW functions.

THE PAD OPTION

It is important to note that annotations do not reserve any space on the graph – they simply draw on top of (or behind) what exists in the graph. There might be times where you want to reserve space in the graph for your annotations, such as a small table outside of the data area. For SAS 9.3, a new option called PAD was added to SGPLOT, SGPANEL, and SGSCATTER to enable you to add padding around the outside of the graph. You can add padding equally to all sides, or you can control the padding separately for each side. The values can be specified with dimensions such as inches, percent, or pixels. Figure 2 shows the effect of specifying padding for all sides versus one side. You will see this option used in a number of the following examples.



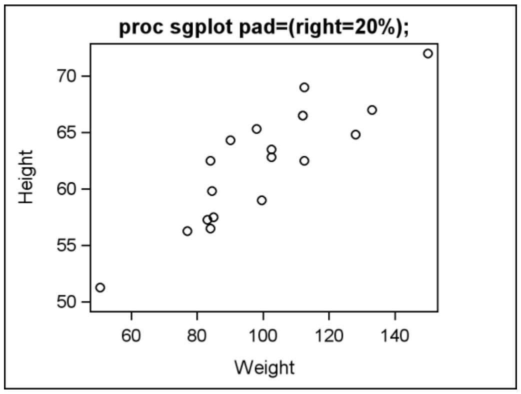


Figure 2. The PAD Option in SGPLOT

FUNCTIONS

The following list contains the currently supported SG annotation functions for SAS® 9.3:

Annotate Function	Description
Text	Draw a text string on the graph.
Textcont	Continue a text string from the Text function. This function is typically used for rich text situations where you want to change the attributes of the text somewhere in the string.
Image	Draw an image on the graph.
Line	Draw a line on the graph.
Arrow	Draw a line with an arrowhead on the graph.
Rectangle	Draw a square or rectangle on the graph.
Oval	Draw a circle or oval on the graph.
Polygon	Draw a closed polygon on the graph. The last point is automatically connected to the first point.
Polyline	Draw a multi-line figure on the graph.
Polycont	Continue a polygon or polyline. The Polygon and Polyline functions specify the starting point for the figure. The Polycont function adds an additional point per function call.

Most functions are self-contained, meaning that they can be drawn using the information in one observation without any dependence on another function call. The two main exceptions to this rule are POLYGON and POLYLINE. These functions must be followed by one or more POLYCONT functions to draw anything. The TEXTCONT function can be used to extend a string from the TEXT function, but it is not required to draw a text string.

THE TEXT FUNCTION

The TEXT function is probably one of the most used functions. This function can be used to add a text string anywhere in the graph. The text does not reserve space in the graph, but it can be used in conjunction with the PAD option to write text around the outside of the graph. The key features of this function include:

- text wrapping
- text rotation
- Unicode, superscript, and subscript support
- transparency

Text Example 1: Adding Unicode characters to Tick Values

In the SG procedures, it is possible to add Unicode characters to any string you enter through syntax, including titles, footnotes, legend labels, axis labels, and so on. However, it is not currently possible to modify the axis tick value strings through the syntax. In this example, I want to add a "≤" to the value to create a more compact way of showing the range (Figure 3).

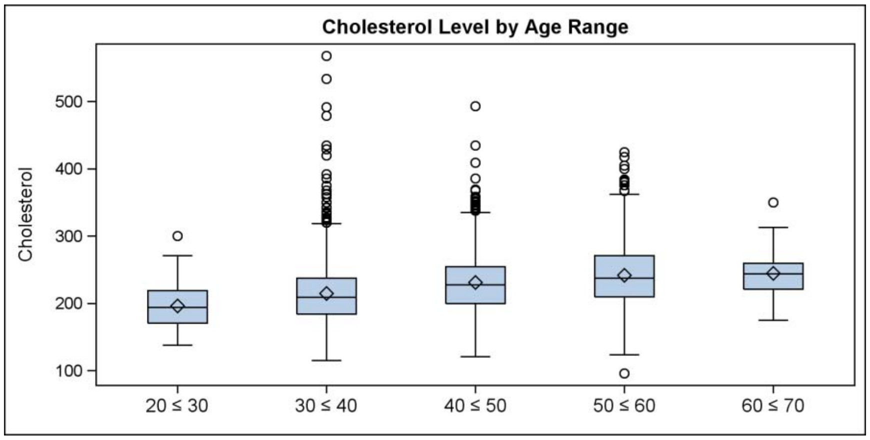


Figure 3. Unicode Characters in Tick Values

```
data anno;
retain function 'text' y1space 'graphpercent' x1space 'datavalue' y1 7 width 15;
input x1 label $ 4-33;
cards;
 21 20 (*ESC*){unicode '2264'x} 30
 31 30 (*ESC*){unicode '2264'x} 40
 41 40 (*ESC*){unicode '2264'x} 50
 51 50 (*ESC*){unicode '2264'x} 60
 61 60 (*ESC*){unicode '2264'x} 70
run;
```

Notice that the Unicode is inserted using the ODS ESCAPECHAR functionality. The entire string definition extends from column 4 to 33. The (*ESC*) starts the escapement and the function is contained within the braces. The text annotation function supports UNICODE, SUP (superscripts), and SUB (subscripts). For the UNICODE function, you can provide the hexadecimal value for the character or use one of the pre-defined names of commonly used characters, such as "alpha" (Greek character) and "hat" (diacritical mark). See the *SAS Graph Template Language: User's Guide* for a complete list of the pre-defined names. Not all fonts contain the pre-defined diacritical marks; therefore, if the mark does not appear, you might need to choose another font.

The X1 values are used to position the strings along the axis using the DATAVALUE drawing space. Notice that this drawing space can be used even though the strings are drawn outside of the data area. This capability is important when you create axis-aligned tables, as you will see in a later example.

This example also adds padding to the bottom of the graph for the strings. The axis label and tick values are turned off in the procedure, which causes the graph to reclaim that space for the data area. To reserve that space for the annotation, you can use the new PAD option.

```
Title1 "Cholesterol Level by Age Range";
proc sgplot data=sashelp.heart sganno=anno pad=(bottom=8%);
  format AgeAtStart agefmt.;
  xaxis display=(nolabel novalues);
  vbox cholesterol / category=AgeAtStart;
run;
```

The PAD option provides the ability to add padding around the outside edge of the graph. If you give the option a single value, such as PAD=8%, then 8% padding will be created around all four sides of the graph. However, you can control the padding for each side by using a suboption of TOP, BOTTOM, LEFT, or RIGHT, as in this example. These suboptions can be used simultaneously to individually control the padding around the graph for more complex annotations.

Text Example 2: Splitting Axis Tick Values

Occasionally, you might have situations where you have long tick values along the X axis of your graph. If the values collide, the default fitting policy for the procedure is to rotate the values, which can cause your data area to become compressed. Changing the policy to STAGGER can help with this, but sometimes the values would appear better if they were split into multiple lines. Currently, there is no axis option for this feature in the procedures; however, there is a way to split the values using annotation.

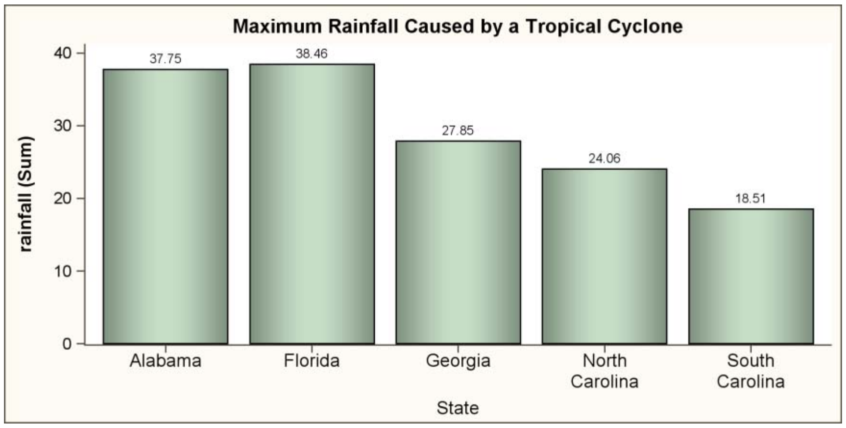


Figure 4. Splitting Tick Values

```
data anno;
  set rainfall (keep=state) end=_last_;
  length x1space $ 11 y1space $ 13 anchor $ 6;
  rename state=xc1;
  retain function "text" x1space "datavalue" y1space "wallpercent" width 15
         justify "center" y1 -1 anchor "top";
  label=state;
  output;
  if (_last_) then do;
    x1space = "wallpercent";
    y1space = "graphpercent";
    anchor="bottom";
    x1=50;
    y1=1;
    label="State";
    output;
  end;
run;
```

The key features for this example are the WIDTH and JUSTIFY columns. The WIDTH column sets the width of the textbox before wrapping occurs. The default width unit is the PERCENT of the X-space, which is the data area in this example. This unit can be changed using the WIDTHUNIT column. The JUSTIFY column sets the justification for each string inside the textbox. In this example, each string resulting from the wrapping is center-justified. The XC1 column (for character X values) is used to position strings along the axis. Also, notice the use of the ANCHOR column. The column sets the anchor point around the text box to be used for positioning. The last record in the annotate data set is used to position the “State” axis label along the midpoint of the wall. As with the previous example, be sure to reserve enough padding along the bottom of the graph to accommodate all of the text.

Text Example 3: Creating an Axis-Aligned Table

The SGPLOT procedure has a very useful feature called an INSET statement that enables you to inset comments or small two-column tables into the data area. However, there are times when it is more useful to have additional statistics in an external table that is aligned with the categorical axis. This table can be created using a technique similar to the previous two examples.

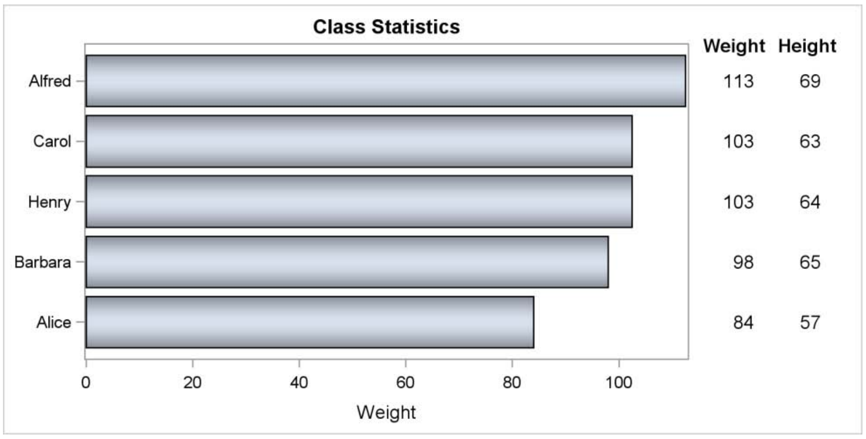


Figure 5. Axis-Aligned Table

```
data anno;
  set sashelp.class (obs=5 keep=name weight height) end=_last_;
  length x1space $ 11 y1space $ 11 anchor $ 8 label $ 6;
  retain function 'text' x1space 'wallpercent' y1space 'datavalue' anchor 'right';
  yc1=name;
  x1=111;
  label=put(weight, F3.0);
  output;
  x1=122;
  label=put(height, F3.0);
  output;
  if (_last_) then do; /* Add the column headers */
    y1space = 'wallpercent';
    width = 20; /* make the textbox wider to prevent wrapping */
    anchor = 'top';
    textweight='bold';
    y1 = 103; /* Slightly above the wall */
    x1=107.5;
    label="Weight";
    output;
    x1=119.5;
    label="Height";
    output;
  end;
run;

ods graphics / reset width=6in height=3in;
Title "Class Statistics";
proc sgplot data=sashelp.class (obs=5) sganno=anno pad=(right=40%);
  yaxis display=(nolabel);
  hbar name / response=weight nostatlabel categoryorder=respdesc dataskin=pressed;
run;
```

This example uses a new option in the SG procedures called CATEGORYORDER to sort the bar chart by the response values in descending order. If I were to remove that option, the Y axis values would be sorted differently and the annotated values would move with them. This occurs because the values are bound to the DATAVALUE space. Therefore, if the data space values move, the annotations move with them.

Notice that the X position of the table values is bound to the wall area, even though the values are outside of the wall. Using this technique binds the table to the right edge of the wall so that, if the wall moves in a subsequent procedure run, the table moves with it. The last observations are used to create the column headings.

Text Example 4: Creating a Confidential Plot

This example highlights the use of text rotation and transparency to create a “CONFIDENTIAL” watermark on a graph. This example uses PROC SGPANEL to create a product sales chart sorted by sales for each year. No drawing space information is needed in the data set because text is drawn in the center of the GRAPHPERCENT space by default.

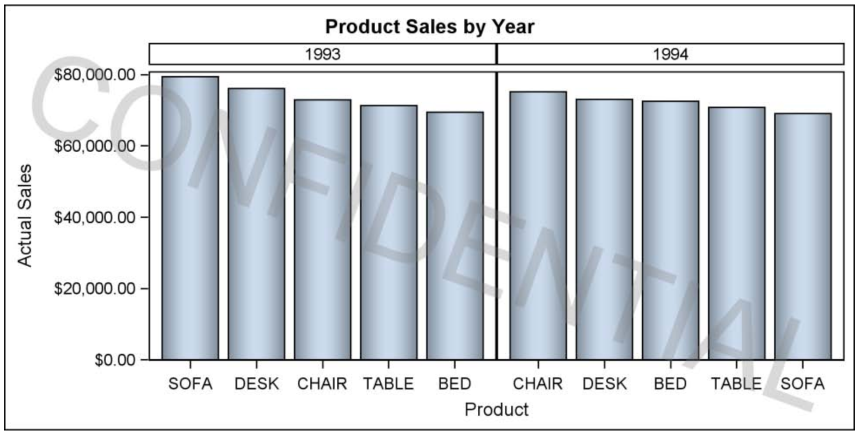


Figure 6. Watermarking a Graph

```
data anno;
  retain function "text" label "CONFIDENTIAL" textcolor "gray" justify "center"
         textsize 64 transparency .70 rotate -20 width 200;
run;

ods graphics / reset width=6in height=3in;
Title1 "Product Sales by Year";
proc sgpanel data=sashelp.prdsale sganno=anno;
   panelby year / layout=columnlattice novarname uniscale=row;
   vbar product / response=actual categoryorder=respdesc dataskin=pressed;
run;
```

The TRANSPARENCY column takes a value between 0 (fully opaque) to 1 (fully transparent). Here, the value of 0.70 was used to make the text very faint against the graph but dark enough to be legible.

The ROTATE column takes a positive or negative number of degrees, and angles the baseline based on that value. There is currently no way to angle the characters independent of the baseline.

Text Example 5: Creating a Forest Plot

This final text example uses a number of the techniques already discussed to create a forest plot using PROC SGPLOT. The procedure does not do the meta-analysis; it simply plots the results. Annotations are used in three areas of the graph:

- the Y axis. The axis label and tick values are disabled, and the tick values are annotated. The purpose is to left-justify the tick values, which you cannot do through axis options.
- the statistics table on the right side of the graph.
- the two effects labels along the X axis.

One way to simplify the table’s creation is to use the DATA step to format the three table values into one string, and use that string in the annotation:

```
Summary = strip(put(OddsRatio,5.3)) || "   (" || strip(put(LowerCL,5.3)) || ", " ||
          strip(put(UpperCL,5.3)) || ")";
```

Because the “no effect” value is in the center of the X axis, I positioned the effect labels by setting the anchor value to BOTTOM and setting the X position to be first 25% and then 75% of the wall area.

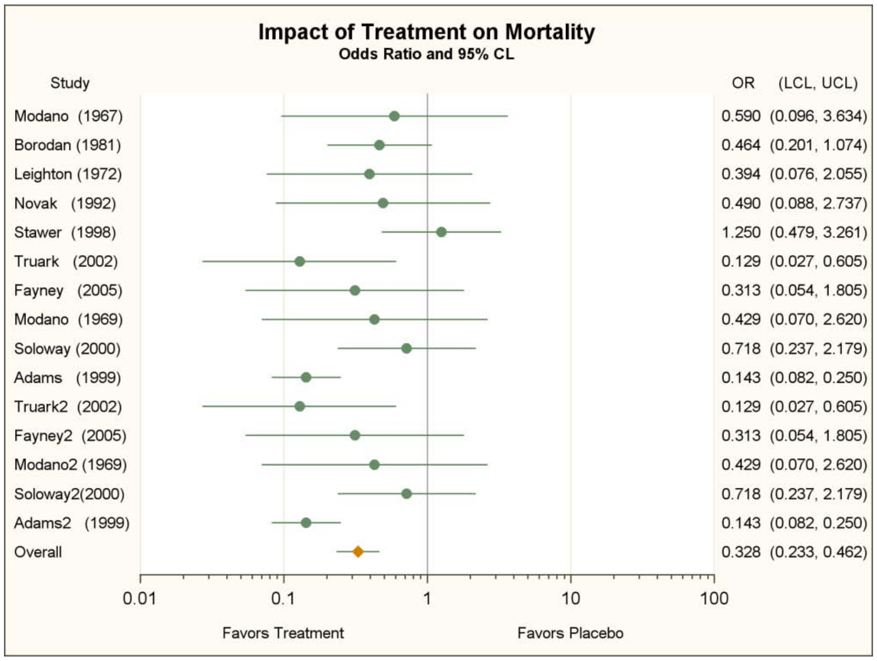


Figure 7. Forest Plot Using PROC SGPLOT

```
data anno;
  set labels end=_last_;
  length label $ 50 anchor $ 6 y1space $ 13 x1space $ 12;
  retain function "text" y1space "datavalue" width 50 anchor "left" textsize 8;
  /* Study and statistics text */
  yc1=study;
  label=study;
  x1space="graphpercent";
  x1=1;
  output;
  label=summary;
  x1space="wallpercent";
  x1=101;
  output;
  if (_last_) then do;
    /* Add the effect labels */
    yc1=" ";
    x1space="wallpercent";
    y1space="graphpercent";
    anchor="bottom";
    y1=2;
    x1=25;
    label="Favors Treatment";
    output;
    x1=75;
    label="Favors Placebo";
    output;
    /* Add the column headers */
    y1space="wallpercent";
    x1space="wallpercent";
    y1=100;
    x1=-12;
    label="Study";
    output;
    x1=105;
    label="OR";
    output;
```

```
      x1=118;
      label="(LCL, UCL)";
      output;
    end;
  run;
```

The SGPLOT code for rendering the graph uses a couple of new features in SAS 9.3: a HIGHLOW plot for drawing the confidence limits and an attribute map to control the appearance of the scatter points based on the group value.

```
  proc sgplot data=meta nocycleattrs noautolegend sganno=anno dattrmap=attrmap
              pad=(left=18% right=22% bottom=7%);
    yaxis reverse display=none;
    xaxis type=log logbase=10 min=.01 max=100 minor display=(nolabel)
          offsetmin=0 offsetmax=0;
    refline 1 / axis=x;
    refline .01 .1 10 100 / axis=x lineattrs=GraphGridLines;
    highlow y=study high=UpperCL low=LowerCL;
    scatter y=study x=oddsratio / attrid=a group=study_type;
  run;
```

THE TEXTCONT FUNCTION

The TEXTCONT function is used primarily to create rich text strings by changing the text attributes. This function is not necessary to use Unicode characters, superscripts, or subscripts, as seen in the first TEXT function example. Once you start using the TEXTCONT function, you must continue to use it until you complete the string. If any other function type occurs in the data set, the string is closed and drawn. See example 1 in the "LINE and ARROW Function" section for an example of this function.

THE IMAGE FUNCTION

The IMAGE function is used to add images to the graph. These images can be positioned anywhere on the graph, and can even be used instead of scatter markers to represent plot points. Images can also be tiled or stretched on the background of the graph by using the LAYER column with a value of BACK. Note that the wall area will obscure any images positioned on the background.

Image Example 1: Adding a Company Logo

A common need for image annotation is to add a company logo to the graph. Because no space is reserved for the image, you will normally need to add a little padding to prevent axis collisions. This data set is simple enough that it should be reusable by all of the graphs in your report.

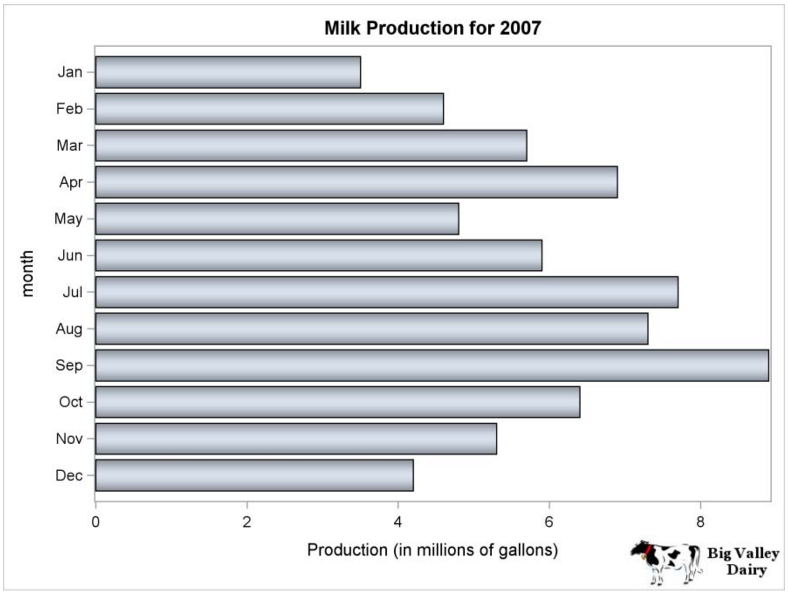


Figure 8. Company Logo in a Bar Chart

```
data anno;
  retain function "Image" anchor "bottomright" x1 100 y1 0.5 width 20
         drawspace "graphpercent" Image "logo.png";
run;

Title "Milk Production for 2007";
proc sgplot data=milk_production pad=(bottom=5%) sganno=anno;
  yaxis discreteorder=data;
  hbar month / response=production dataskin=pressed;
run;
```

In this example, the WIDTH is set to 20% of the graph area. When only one dimension is set on the image, the other dimension is automatically adjusted to maintain the original aspect of the image. You can override this behavior by specifying both WIDTH and HEIGHT. The IMAGESCALE column can also be used to control this behavior. The default value is FIT, which means, “Fit to the bounding box”. This default value was used for the image in Figure 8. The FITWIDTH and FITHEIGHT values always maintain the aspect of the image, ignoring the height and width of the bounding box, respectively. The TILE value will be demonstrated in the next example.

Image Example 2: Putting Images on the Background

An interesting image technique to use in graphs is to put a watermark image on the background. In this example, a small image is tiled on the background with high transparency so that it is not distracting. The LAYER column is used to move the image to the background, the IMAGESCALE column is used to tile the image, and the TRANSPARENCY column is used to fade the image into the background.

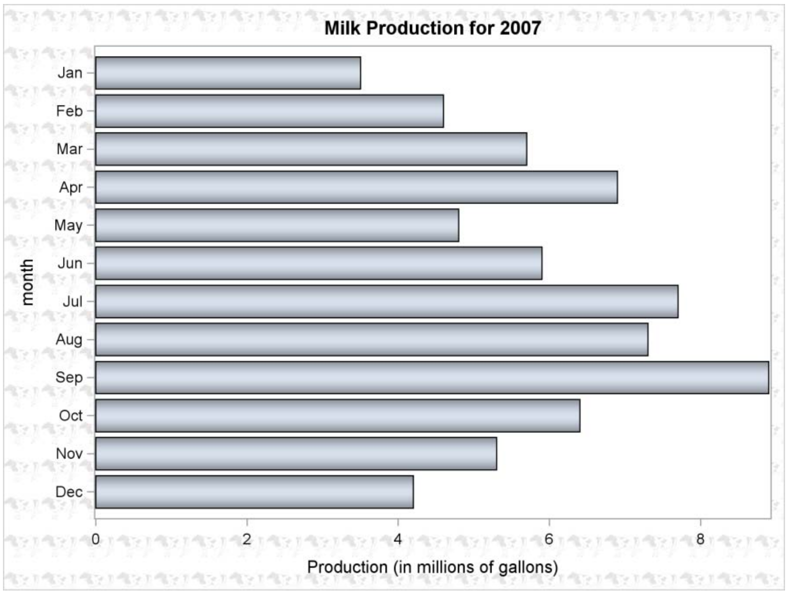


Figure 9. Watermark Images on the Graph Background

```
data anno;
retain function "Image" anchor "bottomright" x1 100 y1 0.5 width 640 height 480
       widthunit 'pixel' heightunit 'pixel' imagescale 'tile'
       drawspace "graphpercent" transparency 0.9 layer 'back' Image "cow.png";
run;

Title "Milk Production for 2007";
proc sgplot data=milk_production sganno=anno;
  yaxis discreteorder=data;
  hbar month / response=production dataskin=pressed;
run;
```

Image Example 3: Using Images as Data Points

Another creative use of images is to use them as scatter markers in place of the standard markers. These images can be used to clearly identify data in the plot without the need for a legend. The images are positioned using the DATAVALUE drawing space, and the images are sized down using the WIDTH column. A WIDTHUNIT value of PIXEL is used in this example to give finer control over the marker size. An ANCHOR value of CENTER is also used to center the image directly on the data point. When using images as markers, make sure that the white space around the core of the image is set to be transparent (you can do this in an image editor). That way, any plot lines used to connect the points will not appear to be clipped around the image.

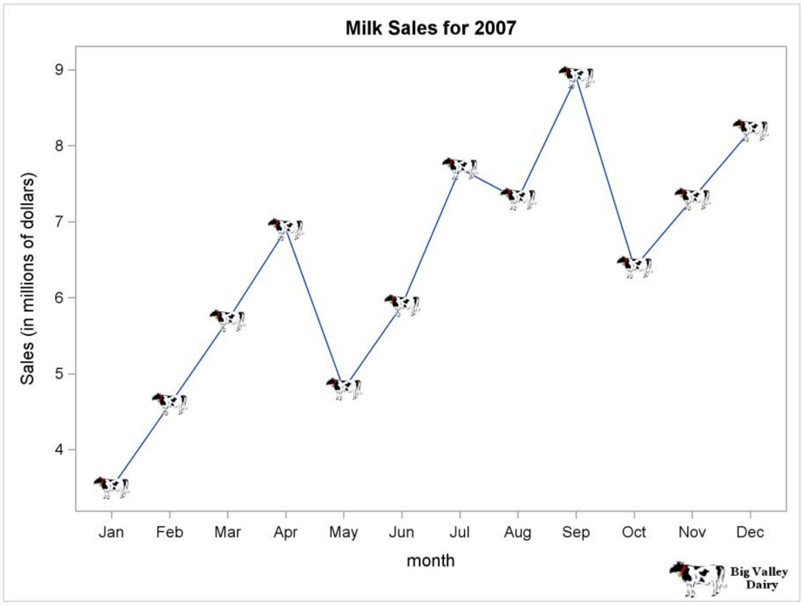


Figure 10. Images as Data Points

```
data anno;
  set milk_sales end=_last_;
  length anchor $ 11 drawspace $ 12 Image $ 76;
  retain function "Image" anchor "center" width 30 widthunit "pixel"
  drawspace "datavalue" Image "cow.png";
  rename month=xc1 sales=y1;
  output;
  if (_last_) then do; /* Add company logo */
    function="Image";
    anchor="bottomright";
    sales=0.5;
    x1=99;
    month=" ";
    width=100;
    drawspace="graphpercent";
    Image="logo.png";
    output;
  end;
run;

Title "Milk Sales for 2007";
proc sgplot data=milk_sales sganno=anno pad=(bottom=5%);
  xaxis offsetmin=0.05 offsetmax=0.05 discreteorder=data;
  yaxis offsetmin=0.05 offsetmax=0.05;
  vline month / response=sales;
run;
```

Image Example 4: Using Images as Curve Labels

Curve labels are a great way to identify plot lines in the graph without the need for a legend, provided that the lines are not too crowded. You can use images as an alternative to text labels to provide more clarity to your audience. The key to this example is to use the last observation (in the case of overlaid lines) or the last observation for each group to position the images. Given the latter case, I used a WHERE clause to collect the observations at the last X axis value. Because the lines extend to the end of the axis, I can use the same X position for all of the images and use only the Y data value to position the images vertically. I used DATAPERCENT for the X position so that I could position the image slightly outside of the data area (102%). I also used an axis offset in the SGPLOT procedure to give extra space at the end of the axis for the images.

Another important note about this graph is the interaction between the company logo and the footnote. I could not use the standard FOOTNOTE statement for the footnote because the padding for the logo would have pushed the footnote up from the bottom of the graph. Therefore, I added the footnote text to the annotation at the same time that I added the company logo so that it would draw in the padded area.

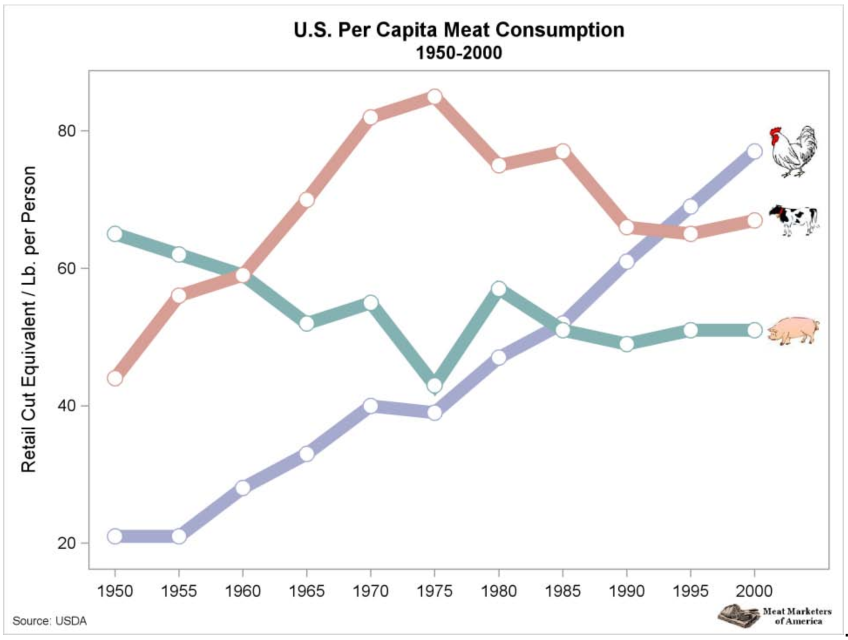


Figure 11. Images as Curve Labels

```
data anno;
  length x1space $ 13 y1space $ 13 anchor $ 11;
  set meat_consumption (where=(year='01jan2000'd)) end=_last_;
  retain anchor 'left' y1space 'datavalue' x1space 'datapercent' width 40
         widthunit 'pixel' function 'image' x1 102;
  y1 = chicken;
  image = "chicken.jpg";
  output;
  y1 = beef;
  image = "cow.jpg";
  output;
  y1 = pork;
  image = "pig.jpg";
  output;
  if (_last_) then do;
     x1space = "graphpercent";
     y1space = "graphpercent";
     anchor = "bottomright";
     x1 = 99;
     y1 = 1;
     width=90;
     image = "Logo.png";
```

```
          output;
          function = "text";
          anchor = "bottomleft";
          x1 = 1;
          width=150;
          textsize = 6;
          label = "Source: USDA";
          output;
       end;
    run;

    Title1 "U.S. Per Capita Meat Consumption";
    title2 "1950-2000";

    /* Overlay white scatter points for the "empty" marker appearance */
    proc sgplot data=meat_consumption noautolegend sganno=anno pad=(bottom=5%);
    xaxis display=(nolabel) offsetmax=0.1;
    yaxis label="Retail Cut Equivalent / Lb. per Person";
    series x=year y=chicken / markers
           markerattrs=(symbol=circlefilled size=12px color=CXa5aace)
           lineattrs=(thickness=10 color=CXa5aace) transparency=0;
    scatter x=year y=chicken / markerattrs=(symbol=circlefilled size=10px color=white);
    series x=year y=pork / markers
           markerattrs=(symbol=circlefilled  size=12px color=CX84b2b3)
           lineattrs=(thickness=10 color=CX84b2b3) transparency=0;
    scatter x=year y=pork / markerattrs=(symbol=circlefilled size=10px color=white);
    series x=year y=beef / markers
           markerattrs=(symbol=circlefilled  size=12px color=CXd69e94)
           lineattrs=(thickness=10 color=CXd69e94) transparency=0;
    scatter x=year y=beef / markerattrs=(symbol=circlefilled size=10px color=white);
    run;
```

THE LINE AND ARROW FUNCTIONS

The LINE and ARROW functions are very similar in their definition and usage. The main difference between them is that the ARROW function supports an additional set of columns that lets you modify the arrowhead behavior. Both functions require the end points to be defined, which means that the X1, Y1, X2, and Y2 columns (or their character counterparts) must be used in the data set.

The three columns used to control the arrowhead behavior are DIRECTION, SCALE, and SHAPE. The DIRECTION column is used to specify which end (or both ends) of the line has or have the arrowhead. The SCALE column is used as a scale factor for the arrowhead size. The SHAPE column is used to specify the shape of the arrowhead. The valid shapes are in Figure 12:

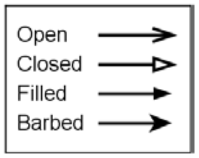


Figure 12. Arrowhead Shapes

Arrow Example 1: Adding Arrow Annotations for Emphasis

Adding arrows to your graph is a great way to emphasize important items in your data. This example adds four arrows leading to the data points from a single comment. The annotation in this example can be recreated in SAS 9.2 using a VECTOR plot (for the arrows) and a SCATTER plot (for the label, using the MARKERCHAR option); however, the process for creating the graph requires you to define all locations in data value coordinates and merge the annotation data with the original data set. The annotation facility makes adding the arrows and label much simpler.

This example also contains a usage of the TEXTCONT function. The label is drawn as one continuous string; however, the string is broken up across a TEXT and two TEXTCONT functions so that the word “VERY” can be made red while the rest of the string is black.

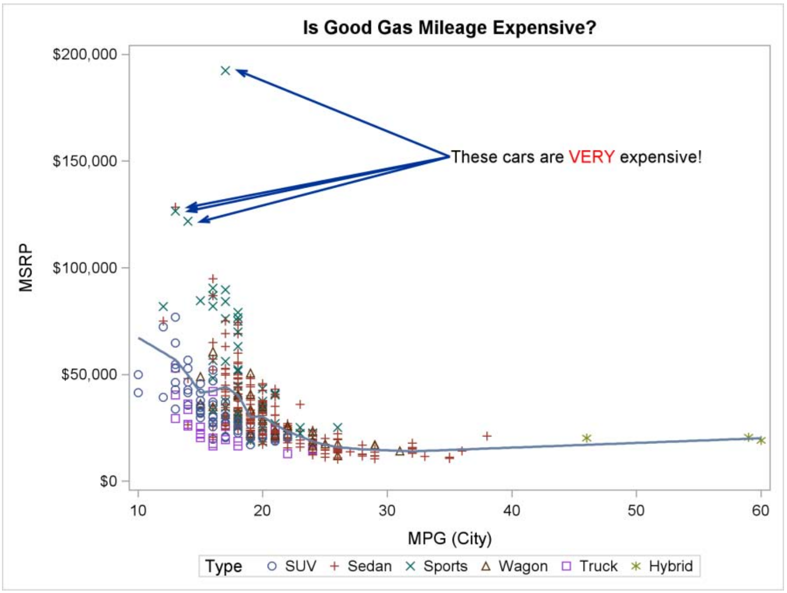


Figure 13. Adding Arrows for Emphasis

```
  data expensive;
    set sashelp.cars;
    where msrp >= 100000;
  run;

  data anno;
    set expensive (keep=msrp mpg_city) end=_last_;
    length x1space $ 11 y1space $ 11 function $ 8 textcolor $ 5;
    retain function "Arrow" x1space "datavalue" y1space "datavalue"
    x2space "wallpercent" y2space "wallpercent" x2 50 y2 75 direction "in" scale 0.5;
    rename msrp=y1 mpg_city=x1;
    mpg_city = mpg_city + 1; /* slightly to the right of the scatter point */
    output;
    if (_last_) then do;
      function = "text";
      x1space = "wallpercent";
      y1space = "wallpercent";
      mpg_city=50;
      msrp=75;
      anchor="left";
      width=50;
      label="These cars are";
      output;
      function = "textcont";
      label=" VERY";
      textcolor="red";
      output;
      label=" expensive!";
      textcolor="black";
      output;
    end;
  run;

  Title "Is Good Gas Mileage Expensive?";
  proc sgplot data=sashelp.cars sganno=anno;
```

```
    scatter x=mpg_city y=msrp / group=type;
    loess x=mpg_city y=msrp / nomarkers;
  run;
```

THE RECTANGLE AND OVAL FUNCTIONS

Although the RECTANGLE and OVAL functions are very different primitives, their definitions are very similar. A single point defines the location, and the WIDTH and HEIGHT columns define the size and aspect of the primitive. To create squares and circles, set the width and height to be the same value. Use the ANCHOR column to specify the position point used to anchor the primitive. Both primitives can be filled or unfilled, and both primitives support the FILLTRANSPARENCY column in addition to the TRANSPARENCY column. The FILLTRANSPARENCY column is used to add transparency to the filled area only, while the outline remains fully opaque.

One unique option for the RECTANGLE function is the CORNERRADIUS column. This column can be used to round the corners of a rectangle. The valid range for this column is from 0.0-1.0, with a default of 0.0 (no rounding).

Oval Example 1: Creating a Bubble Legend

This example uses the new bubble plot that is added to the SG procedures in SAS 9.3. The bubble labels reflect the Y axis value by default (the data label variable can be assigned), but you might also want a legend showing the values for bubble sizes. You can make a bubble legend using annotate.

The key to making the legend sizes synchronize with the plot is using the BRADIUSMIN and BRADIUSMAX options in the BUBBLE statement of the procedure. You can use these options to set the minimum and maximum size of the bubbles. You then use those sizes to create the bubbles in the legend for the minimum and maximum data values.

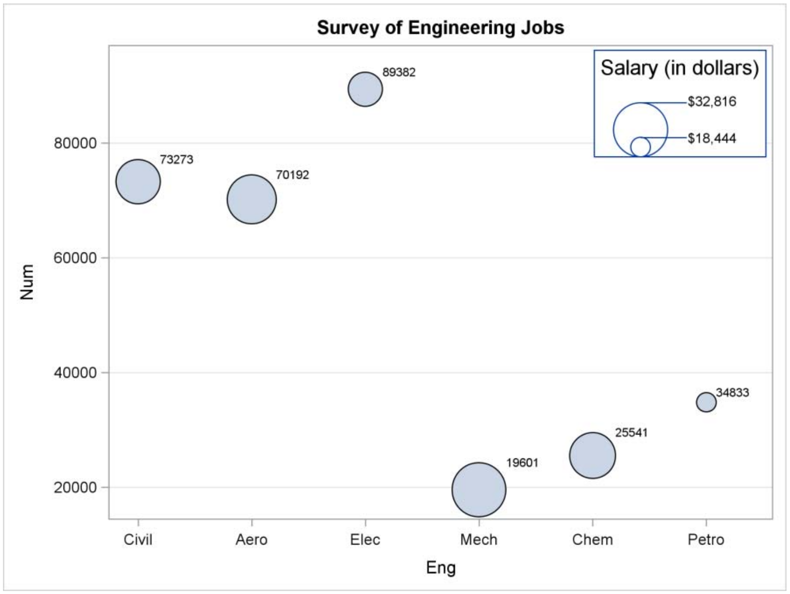


Figure 14. Adding a Bubble Legend

```
  data anno;
    retain drawspace "wallpercent" widthunit "pixel" heightunit "pixel"
           linethickness 1 textsize 8;
    length function $ 9;
    input function $ x1 y1 width height x2 y2 textsize anchor $ label $ 48-66
          display $ 68-74 fillcolor $ 76-80;
    cards;
    Rectangle  86 76.5  140  87  .     . 12 bottom                     all     white
    Oval       80 76.5   44  44  .     . 12 bottom                     outline
    Oval       80 76.5   16  16  .     . 12 bottom                     outline
    Line       80 87.9    .   . 87  87.9 12 bottom
    Line       80 80.5    .   . 87  80.5 12 bottom
    Text       86 98    140   .  .     . 12 top    Salary (in dollars)
    Text        87 88.1  140    .  .      .  8 left   $32,816
    Text        87 80.4  140    .  .      .  8 left   $18,444
  ;
  run;

  Title1 "Survey of Engineering Jobs";
  proc sgplot data=jobs sganno=anno;
    yaxis grid;
    bubble y=Num x=Eng size=Dollars / datalabel bradiusmax=22px bradiusmin=8px;
  run;
```

Notice that the grid line does not go through the bubble legend. The DISPLAY column sets both the fill and the outline to be enabled. The FILLCOLOR column is set to be white so that anything behind the legend is hidden. The DISPLAY value is set back to outline for the ovals.

THE POLYGON, POLYLINE, AND POLYCONT FUNCTIONS

There are two main differences between the POLYGON and POLYLINE functions:

- The POLYGON function always draws a closed polygon, that is, the last point is always connected to the first point. The POLYLINE function can be used to draw open or closed polygons.
- The POLYGON function annotation can be filled with a color. The POLYLINE function annotation cannot be filled.

The POLYGON or POLYLINE functions specify the starting point for the polygon or polyline, but you must specify subsequent POLYCONT functions for each point. The only attributes you can change for the polygon in the POLYCONT function are the drawing spaces of the current coordinates.

Polyline Example 1: Showing Relationship in a Bar Chart

This example uses annotation to show the resulting unit volume when two car companies merged. Instead of using arrows only, this example uses polylines to create a structure that makes the connections easier to see.

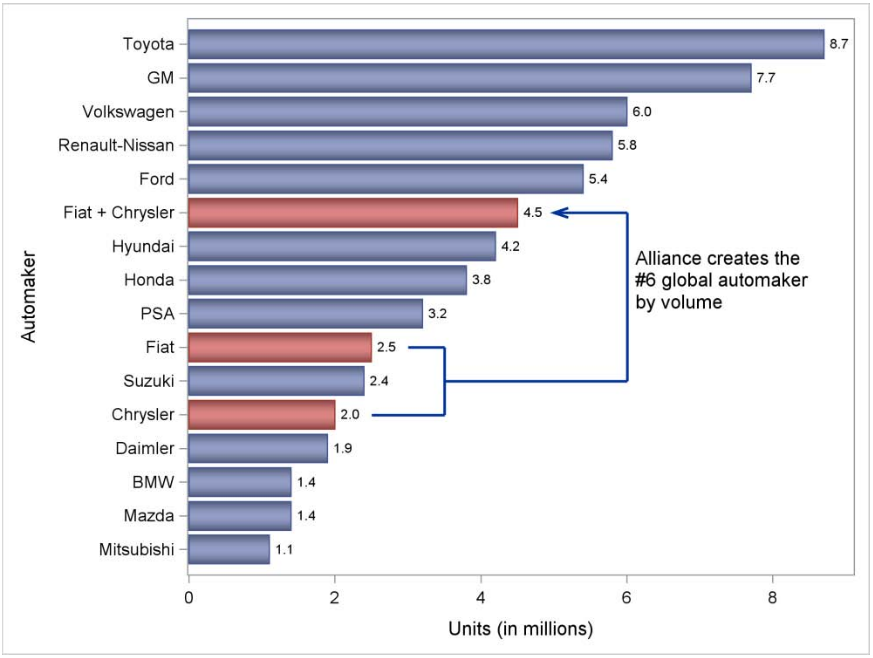


Figure 15. Showing Relationship in a Bar Chart

```
  data anno;
    length yc1 $ 15;
    retain drawspace "datavalue";
    function="polyline";
    yc1="Chrysler";
    x1=2.5;
    output;
    function="polycont";
    x1=3.5;
    output;
    yc1="Fiat";
    output;
    x1=3.0;
    output;
    function="polyline";
    yc1="Suzuki";
    x1=3.5;
    output;
    function="polycont";
    x1=6.0;
    output;
    yc1="Fiat + Chrysler";
    output;
    function="arrow";
    x2=5.0;
    yc2="Fiat + Chrysler";
    output;
    function="text";
    yc1="Honda";
    x1=6.1;
    anchor="Left";
    width=30;
    label="Alliance creates the #6 global automaker by volume";
    output;
  run;

  proc sgplot data=autos noautolegend sganno=anno;
     hbarparm category=Automaker response=million_units / datalabel group=colorvar
              dataskin=pressed;
  run;
```

CONCLUSION

As mentioned in the introduction, there are currently 45 reserved column names in this annotation facility. Although every possibility could not be covered in this paper, hopefully the included examples will give you some idea of what is possible. Be sure to consult the documentation for a full list of the options available for each function. While it might be possible to create entire graphs using this facility, the intent of this facility is to enable users to add customizations not covered by existing procedure statements and options. For this first pre-production release, you will find that there is functionality in the SAS/GRAPH® annotation facility not contained in this SG procedure facility (and vice versa). As this new facility develops, it is important that you give us feedback so that the most needed features are incorporated into the facility.

ACKNOWLEDGMENTS

I want to thank Sanjay Matange, Susan Schwartz, and Melisa Turner for their valuable input.

REFERENCES

Heath, Dan. 2009. “Secrets of the SG Procedures.” *Proceedings of the SAS Global Forum 2009 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings09/324-2009.pdf .

CONTACT INFORMATION

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

Dan Heath
500 SAS Campus Drive

SAS Institute Inc.
Cary, NC 27513
Dan.Heath@sas.com