Paper 382-2013

GTL to the Rescue!

Lelia McConnell, SAS Institute Inc.

ABSTRACT

You just produced some graphs using the SGPLOT and SGPANEL procedures. Now you want to modify the structure of your graphs in order to make them more meaningful. You are looking for options that enable you to split the axis values across multiple lines, add a table under a graph, or create a template with which you can conditionally execute statements and dynamically assign variables. However, you cannot find any options within the procedure that enable you to put these final touches on your graphs. When all seems hopeless, ODS Graphics Template Language (GTL) comes to the rescue!

INTRODUCTION

Beginning with SAS® 9.3, the SAS/GRAPH® Statistical Graphics (SG) procedures and Graph Template Language (GTL) have moved to the Base SAS® product. These procedures, which now are called *ODS Graphics procedures*, include PROC SGPLOT, PROC SGPANEL, PROC SGSCATTER, PROC SGDESIGN, and PROC SGRENDER. This move is beneficial because you no longer need SAS/GRAPH software in order to use the SG procedures. Now, all users have access to the power of the ODS Statistical Graphics family of procedures.

GTL is the backbone of the SG procedures. However, not all of its functionality is apparent through the procedures. By using GTL directly, you have the ability to create graphs that you cannot create with standard SG procedures.

SAS Technical Support has many opportunities to help SAS customers use GTL to customize their graphics output. This paper is a collection of some of these examples. The following topics show you how to use SAS 9.3 template code to tap into the hidden power of GTL and get results that are typically unattainable from an SG procedure:

- Overlaying a Scatter Plot and Bar Chart
- Adding Left and Right Vertical Axes
- Defining Custom Tool Tips for Your Graph
- Splitting the Axis Values Across Multiple Lines
- Displaying Unicode Values in Tick Values
- Defining Dynamic Variables and Using the EVAL Expression
- Defining Discrete Attribute Maps
- Adding a Table under a Graph
- Creating a Break in the Axis
- Building Side-by-Side Graphs with Y and Y2 Axes

Note that a basic knowledge of GTL is required in order to apply these techniques.

OVERLAYING A SCATTER PLOT AND BAR CHART

A benefit of producing graphs with ODS Graphics is having the ability to overlay multiple plots in a single cell. This is not always possible though. For example, you might use PROC SGPLOT to overlay a scatter plot and a bar chart that contains multiple response values for one or more categories in the same cell. Here is your code:

```
data simple;
   input Topic Sum Partial;
   datalines;
1 50 24
1 30 10
2 75 10
3 40 37
3 12 19
4 100 75
5 60 40
;
run;
```

```
proc sgplot data=simple;
   vbar topic / response=Sum;
   scatter x=topic y=Partial;
run;
```

However, the graph that you expected is not produced, and the following error message is written to the SAS log:

```
ERROR: Attempting to overlay incompatible plot or chart types.
```

Although PROC SGPLOT supports overlaying multiple plot statements in a single cell, not all plot statements are compatible. For information about the four types of plots that you can create with the SGPLOT and SGPANEL procedures and the compatibility limitations, see the "Concepts: SGPLOT Procedure" section of the *SAS® 9.3 ODS Graphics Procedures Guide, Third Edition* (support.sas.com/documentation/cdl/en/grstatproc/65235/PDF/default/grstatproc.pdf). The compatibility limitations do not exist when you use GTL to overlay a scatter plot and bar chart.

The following program uses GTL to overlay a scatter plot and a bar chart that contains multiple response values for one or more categories successfully in the same cell. The program combines a BARCHART statement and a SCATTERPLOT statement within in the same LAYOUT OVERLAY block to produce the output that is shown in Figure 1.

```
proc template;
define statgraph barscatter;
begingraph;
entrytitle 'Overlay Scatter Plot and Bar Chart';
   layout overlay;
      barchart x=Topic y=Sum;
      scatterplot x=Topic y=Partial;
   endlayout;
endgraph;
end;

proc sgrender data=simple template=barscatter;
run;
```

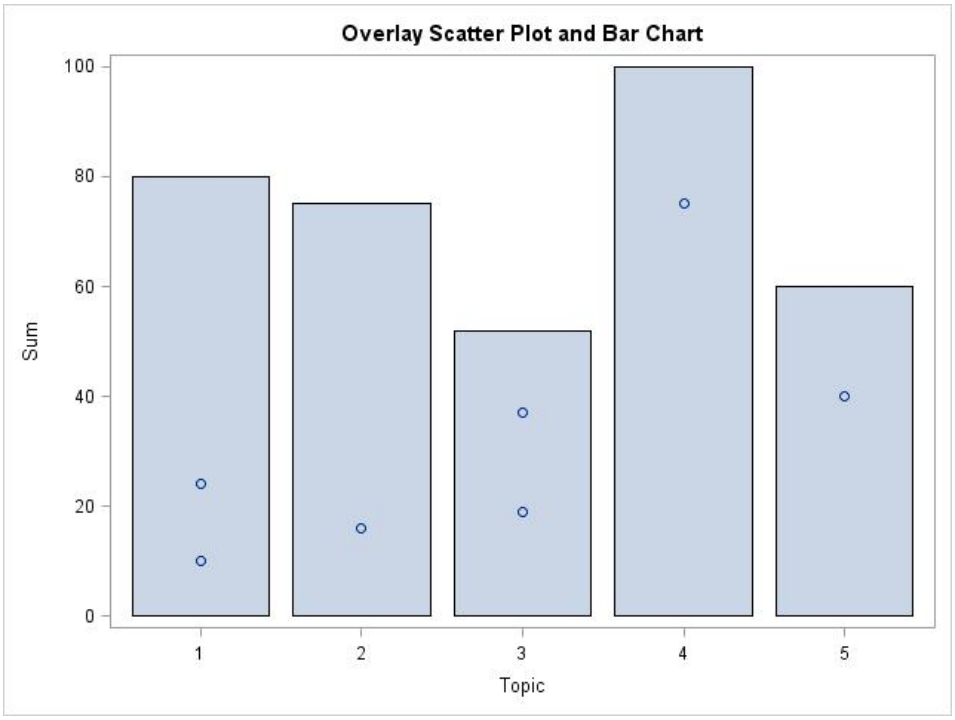

Figure 1. Output That Overlays a Scatter Plot and a Bar Chart in the Same Cell

ADDING LEFT AND RIGHT VERTICAL AXES

In order to give a better indication of the values for particular data points, you might want to show an additional Y axis. Adding a second Y axis is not feasible using any of the SG procedures. However, with GTL, you can plot the data values for the bar chart on the right vertical axis by adding the option YAXIS=Y2 in the BARCHART statement, as shown in the following code:

```
proc template;
   define statgraph vbarscat;
   begingraph;
   entrytitle 'Overlay Bar Chart and Scatter Plot';
      layout overlay /yaxisopts=(linearopts=(viewmin=0 viewmax=80
                      tickvaluesequence=(start=0 end=80 increment=10)));
            barchart x=Topic y=Sum /yaxis=y2;
            scatterplot y=Partial x=Topic;
      endlayout;
   endgraph;
   end;

proc sgrender data=simple template=vbarscat;
run;
```

The output is shown in Figure 2.

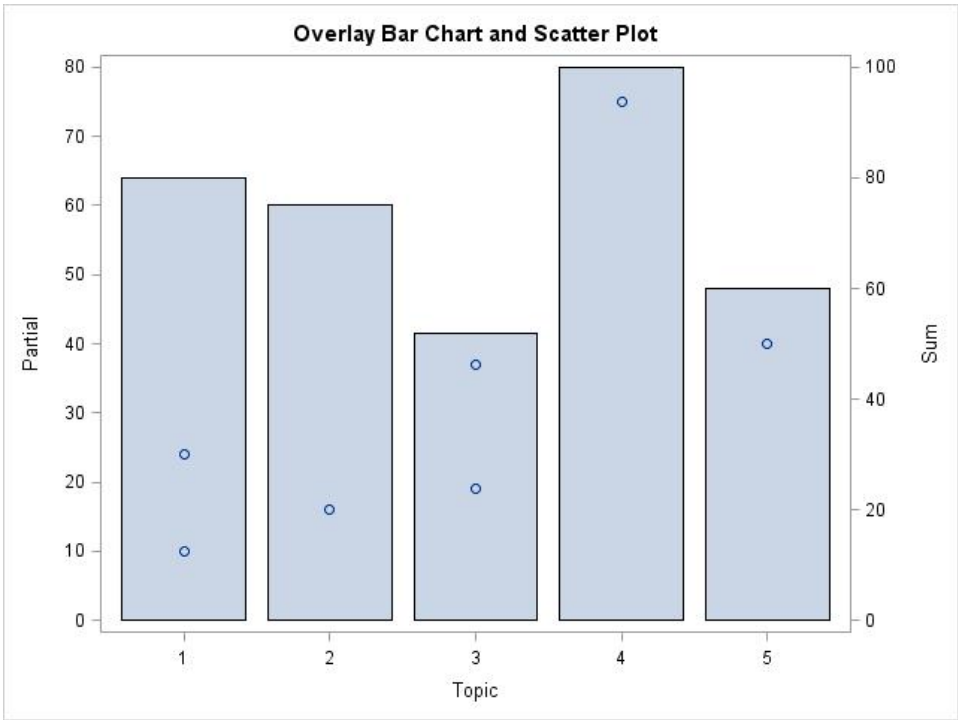

Figure 2. Output Showing an Additional Y Axis

DEFINING CUSTOM TOOL TIPS FOR YOUR GRAPH

You might want to display custom tool tips on your graph so that information about the data is displayed when you hold your mouse pointer over a data location on the graph. The following sample program uses the ROLENAME= and TIP= options in the SCATTERPLOT statement to modify the default tool tips that are displayed. The ROLENAME= option assigns the variables Name and Sex as the tips to be displayed and the TIP= option specifies the order in which the tips are displayed. In order to display the tool tips, it is also necessary to specify the option IMAGEMAP= in the ODS GRAPHICS statement. If this option is not specified, the tool tips are not generated or displayed.

```
proc template;
   define statgraph sgplot;
   begingraph;
      entrytitle 'Customizing Tool Tips with GTL';
      layout overlay;
         scatterplot x=age y=height / primary=true
                                      rolename=(tip1=name tip2=sex)
                                      tip=(tip1 tip2);
      endlayout;
   endgraph;
   end;
run;

ods graphics / imagemap;

proc sgrender data=sashelp.class template=sgplot;
run;
```

The output is shown in Figure 3.

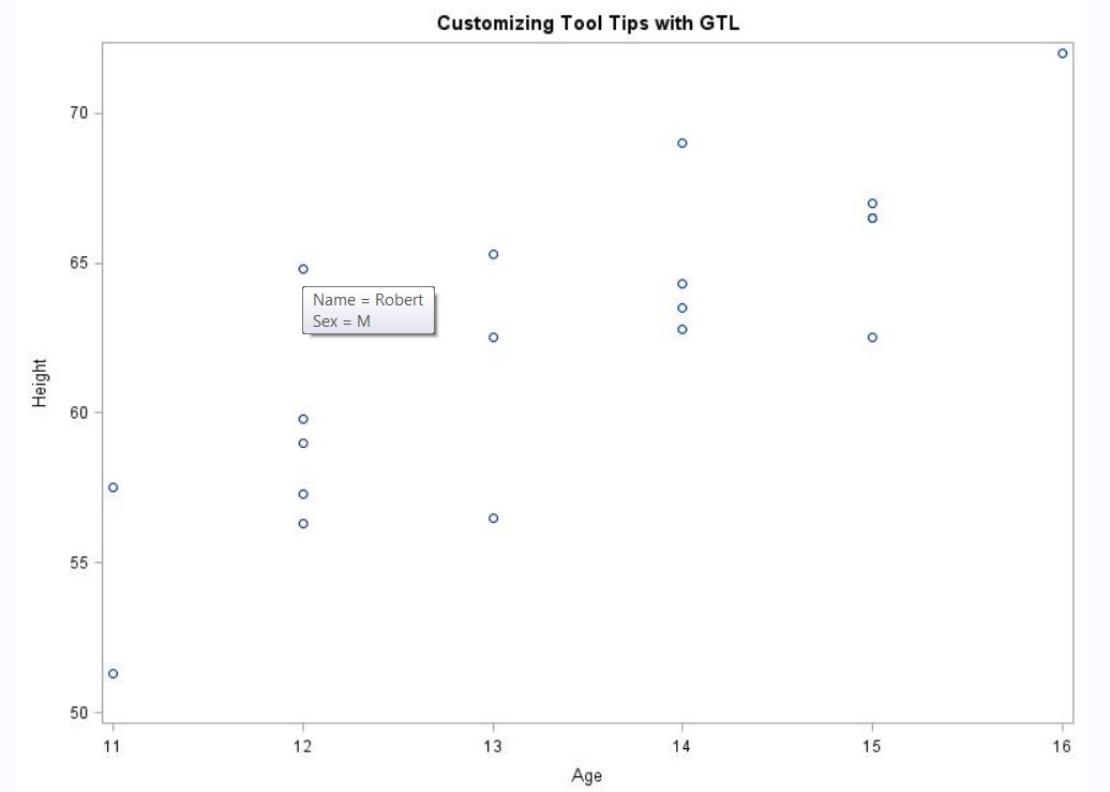

Figure 3. Tool Tips Are Displayed When You Hover over Data Points

SPLITTING AXIS VALUES ACROSS MULTIPLE LINES

Beginning with SAS 9.3 (TS1M0), you can use DRAW statements within GTL to add text, lines, arrow, ovals, circles, polygons, and images to your graphics output. These DRAW statements are referred to as *annotation*. The ability to annotate is preproduction with the SG procedures in SAS 9.3 (TS1M0) and production in SAS 9.3 (TS1M1).

Within annotation, you can use the DRAWTEXT statement to draw text on the graphics output. As with the Annotate facility in SAS/GRAPH, you need to assign several variables in the DRAWTEXT statement in order to describe the location of the text and the text attributes. The X and Y variables define the anchor coordinates for the text. The ANCHOR variable defines the anchor point for the text box. Values for this variable are CENTER, TOPLEFT, TOP,

TOPRIGHT, LEFT, RIGHT, BOTTOMLEFT, BOTTOM, and BOTTOMRIGHT. The variables XSPACE and YSPACE define the drawing space and drawing units for interpreting the X and Y values.

In the following example, XSPACE=DATAVALUE means that each value for X will be interpreted as an actual data value on the graph. Because YSPACE=DATAPERCENT, each value for Y will be interpreted as a percentage of the graphics area. The WIDTH variable defines the width of the text box. The null ENTRYFOOTNOTE statement sets the size of the text to 30 pixels to ensure that there is enough space below the graph for the annotation. The annotation draws the axis values and the axis label. The WHERE statement in PROC SGRENDER subsets Canada and the United States from the original data set. The output is shown in Figure 4.

```
proc template;
   define statgraph drawtext;
   begingraph;
      entrytitle 'Shoe Sales in Canada and the United States';
      entryfootnote ' ' / textattrs=(size=30);
      layout overlay / xaxisopts=(display=(line ticks));
         barchart x=product y=sales / group=region groupdisplay=cluster
                                      name='region' dataskin=sheen;
         drawtext 'Sandal' / x='Sandal' y=-5 anchor=top justify=center
                        xspace=datavalue yspace=datapercent width=15;
         drawtext 'Boot' / x='Boot' y=-5 anchor=top justify=center
                       xspace=datavalue yspace=datapercent width=15;
         drawtext 'Slipper' / x='Slipper' y=-5 anchor=top justify=center
                         xspace=datavalue yspace=datapercent width=15;
         drawtext 'Sport Shoe' / x='Sport Shoe' y=-5 anchor=top
                           justify=center xspace=datavalue
                                         yspace=datapercent width=15;
         drawtext "Men's Dress" / x="Men's Dress" y=-5 anchor=top
                            justify=center xspace=datavalue
                             yspace=datapercent width=15;
         drawtext "Women's Casual" / x="Women's Casual" y=-5 anchor=top
                               justify=center xspace=datavalue
                                              yspace=datapercent width=15;
         drawtext "Women's Dress" / x="Women's Dress" y=-5 anchor=top
            justify=center xspace=datavalue yspace=datapercent width=15;
         drawtext "Men's Casual" / x="Men's Casual" y=-5 anchor=top
            justify=center xspace=datavalue yspace=datapercent width=15;
         drawtext 'Type of Shoe' / x=50 y=2 justify=center
                               xspace=graphpercent
                               yspace=graphpercent width=30;
         discretelegend 'region' / location=inside halign=right valign=top;
      endlayout;
   endgraph;
   end;

   define style styles.barcolor;
   parent=styles.htmlblue;
      class GraphData1 / color=VIOY;
      class GraphData2 / color=BIB;
   end;
run;

proc sort data=sashelp.shoes out=shoes;
   by product;
run;

ods html style=styles.barcolor;

proc sgrender data=shoes template=drawtext;
   where region in('United States' 'Canada');
run;
```

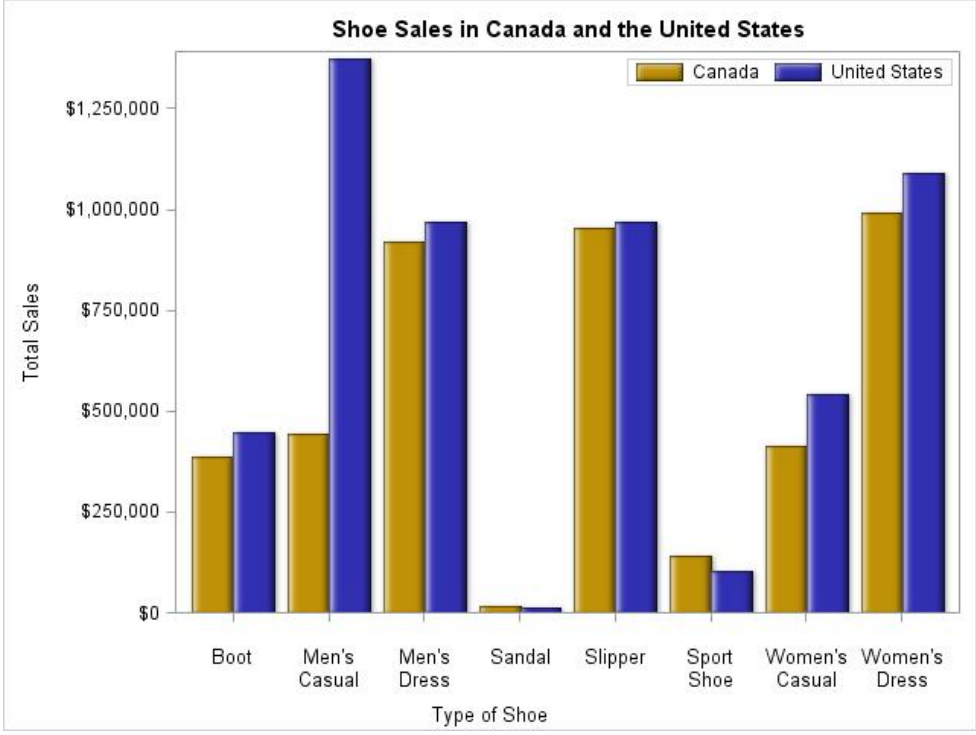

Figure 4. Output Showing Axis Values Split across Multiple Lines

DISPLAYING UNICODE VALUES IN TICK VALUES

Beginning with SAS 9.3, you can use annotation to display Unicode values as part of the tick values. In order to display Unicode values, you need to define a font that supports Unicode values in your style definition. In the following example, the Monotype Sans WT J font is used in the style definition for the style UNIFONTS. This style is referenced with the STYLE option in the ODS destination statement. Unicode values are specified in the form {unicode '2264'x}. The following sample code shows the syntax, and the output is shown in Figure 5:

```
proc template;
   define style unifonts;
   parent=Styles.meadow;
   style Graphfonts from GraphFonts /
      'GraphValueFont' = ("Monotype Sans WT J",12pt)
      'GraphLabelFont' = ("Monotype Sans WT J",12pt)
      'GraphDataFont' = ("Monotype Sans WT J",12pt)
      'GraphTitleFont' = ("Monotype Sans WT J",12pt)
      'GraphFootnoteFont'=("Monotype Sans WT J",12pt)
      'GraphAnnoFont'=("Monotype Sans WT J",12pt);
   end;
run;

proc format;
   value agefmt
      low-39 = "Under 40"
      40-50  = "40 to 50 Years"
      51-70  = "51 to 70 Years"
      71-high = "Over 70";
run;

proc freq data=sashelp.heart noprint ;
   where status = "Dead";
   format ageatdeath agefmt.;
```

```
   tables ageatdeath / out=ages;
run;

proc template;
   define statgraph unicodes;
   begingraph;
   entrytitle ‘Cause of Death’;
      layout overlay / xaxisopts=(display=(line))
                       yaxisopts=(label='count');
         barchart x=ageatdeath /group=deathcause groupdisplay=cluster
               name='bars';
            discretelegend 'bars' /location=inside halign=left valign=top across=1;
            drawtext '>40'  /xspace=datavalue yspace=datavalue y=-6 width=30
                              x='Under 40';
            drawtext '40 ' {unicode '2264'x}' 50'  /xspace=datavalue
                                                   yspace=datavalue y=-6 width=30
                              x='40 to 50 Years';
            drawtext '51 ' {unicode '2264'x}' 70'  /xspace=datavalue
                                                   yspace=datavalue y=-6 width=30
                              x='51 to 70 Years';
            drawtext 'Over 70'  /xspace=datavalue yspace=datavalue y=-6 width=30
                              x='Over 70';
      endlayout;
   endgraph;
   end;

ods html style=unifonts;

   proc sgrender data=sashelp.heart template=unicodes;
      where ageatdeath ne .;
   format ageatdeath agefmt.;
   run;
```

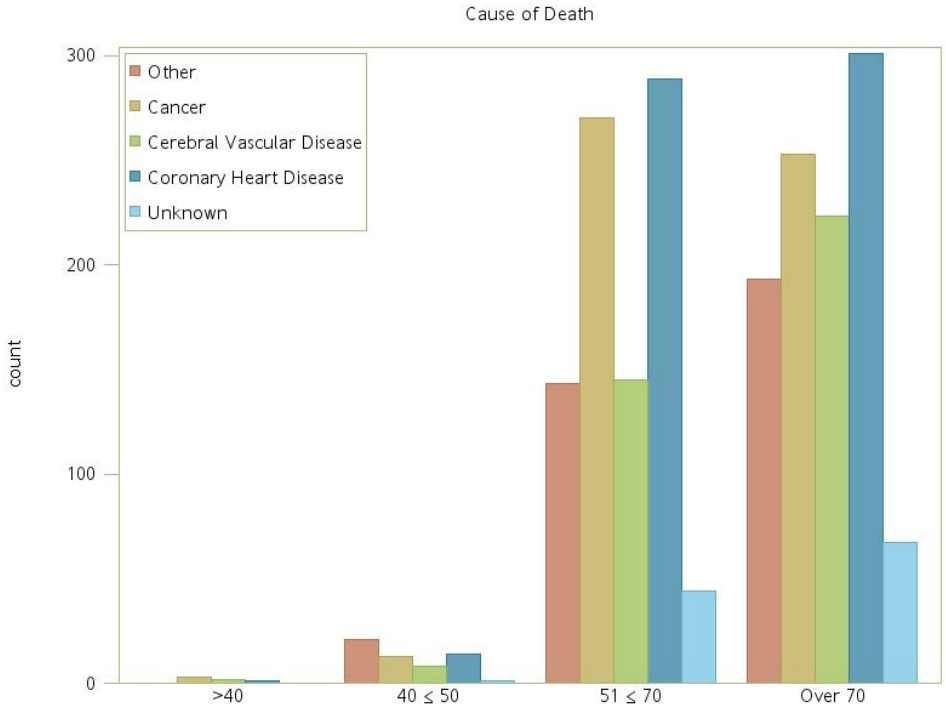

Figure 5. Displaying Unicode Values in Tick Values

DEFINING DYNAMIC VARIABLES AND USING THE EVAL EXPRESSION

Using dynamic variables gives you the flexibility to change variable assignments within the template definition from PROC SGRENDER without recompiling your template definition. You can either define your own dynamic variables with GTL or use any of the special dynamic variables. For more information about special variables, see the "Special Dynamic Variables" section of the *SAS® 9.3 ODS Graphics Procedures Guide, Third Edition* (support.sas.com/documentation/cdl/en/grstatproc/65235/PDF/default/grstatproc.pdf).

You define the special variables in the DYNAMIC statement after the BEGINGRAPH statement within your template definition and later assign values in the DYNAMIC statement in PROC SGRENDER. Special dynamic variables are also defined in the DYNAMIC statement in your template definition, but you do not need to assign a value to the variable in the DYNAMIC statement inside of PROC SGRENDER.

In the following example, the dynamic variable MONEY is defined in the DYNAMIC statement in the template. Using IF logic, you can define a template definition that creates a vertical bar chart if the value of Money is `Sales` and a horizontal bar chart if the value of Money is `Returns`.

Expressions are used to set option values that are constants, columns, or part of the text for some statements. The EVAL expression is specific to GTL and is used in combination with Base SAS and GTL functions. In the example, the EVAL expression defines the X/Y location of the REFERENCELINE and the value that is to be displayed for the curve label. Within the EVAL expression, the MEANS function is used to return the mean value of Returns or Sales. You also format the result using the dollar10.2 format.

Within PROC SGRENDER, the dynamic variable MONEY is assigned a value of `Returns`. The output from the sample code is shown in Figures 6 and 7.

```
proc template;
   define statgraph drawtext;
      begingraph;
        dynamic money _byval_;
        entryfootnote ' ' / textattrs=(size=30);

      if(colname(money)='Sales')
      entrytitle 'Shoe Sales in ' _byval_ ;
      layout overlay/ xaxisopts=(display=(line ticks));
      barchart x=product y=sales/dataskin=sheen group=product fillattrs=GraphData3;
      referenceline y=eval(mean(sales)) / lineattrs=(color=green) curvelabel=
                    eval(putn(mean(sales),'dollar12.2'))
                    curvelabelattrs=(color=green);
         drawtext 'Sandal' / x='Sandal' y=-5 anchor=top justify=center
                    xspace=datavalue yspace=datapercent width=15;
         drawtext 'Boot' / x='Boot' y=-5 anchor=top justify=center xspace=datavalue
                             yspace=datapercent width=15;
         drawtext 'Slipper' / x='Slipper' y=-5 anchor=top justify=center
                           xspace=datavalue yspace=datapercent width=15;
         drawtext 'Sport Shoe' / x='Sport Shoe' y=-5 anchor=top justify=center
                              xspace=datavalue yspace=datapercent width=15;
         drawtext "Men's Dress" / x="Men's Dress" y=-5 anchor=top justify=center
                             xspace=datavalue yspace=datapercent width=15;
         drawtext "Women's Casual" / x="Women's Casual" y=-5 anchor=top
                              justify=center xspace=datavalue
                               yspace=datapercent width=15;
         drawtext "Women's Dress" / x="Women's Dress" y=-5 anchor=top
                              justify=center xspace=datavalue
                               yspace=datapercent width=15;
         drawtext "Men's Casual" / x="Men's Casual" y=-5 anchor=top justify=center
                              xspace=datavalue yspace=datapercent width=15;
         drawtext 'Type of Shoe' / x=50 y=2 justify=center xspace=graphpercent
                              yspace=graphpercent width=30;
      endlayout;
      endif;
      if (colname(money)='Returns')
      entrytitle 'Shoe Returns in ' _byval_ ;
      layout overlay / ;
         barchart x=product y=returns/dataskin=sheen group=prodfill
```

```
         orient=horizontal fillattrs=GraphData3;
            referenceline x=eval(mean(returns)) / lineattrs=(color=red)
                 curvelabel=eval(putn(mean(returns),'dollar12.2'))
                 curvelabelattrs=(color=red);
      endlayout;
      endif;
   endgraph;
   end;
run;

proc sort data=sashelp.shoes out=shoes;
   by region;
run;

proc sgrender data=shoes template=drawtext;
   where region in('United States' 'Canada');
   by region;
   dynamic money='Returns';
run;
```

Figure 6. Output Where the Dynamic Variable MONEY='Returns' and REGION='Canada'

Figure 7. Output Where the Dynamic Variable MONEY='Returns' and REGION='United States'

When you change MONEY='Returns' to MONEY='Sales' and remove 'Canada' from the WHERE statement, you see the sale of shoes for the United States. See Figure 8.

```
proc sgrender data=shoes template=drawtext;
   where region in('United States');
   by region;
   dynamic money='Sales';
run;
```

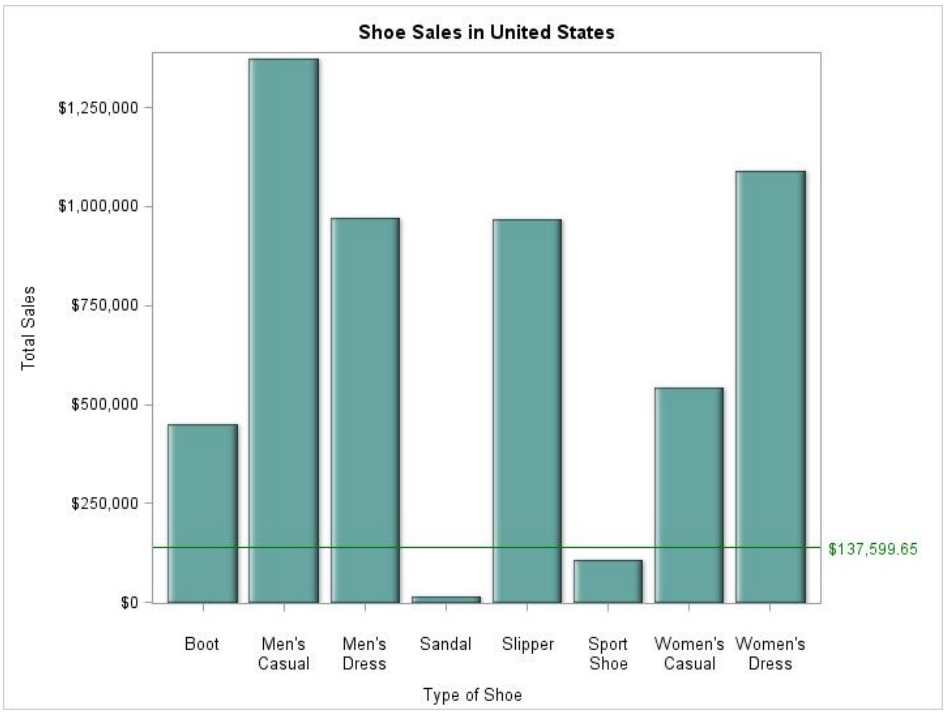

Figure 8. Output Where the Dynamic Variable MONEY='Sales' and REGION='United States'

DEFINING DISCRETE ATTRIBUTE MAPS

You are creating a series of graphs in which all midpoint values are not present for all BY values. You want the fill colors to be consistent across graphs. To do this, you define a discrete attribute map. Defining a discrete attribute map gives you the ability to control the style element attributes for each possible value of a variable across graphs.

In the following example, the discrete attribute map assigns the fill color for each possible value of the variable STUDENT. This ensures that the fill color for each student value is the same even if all possible values are not present for each value of the BY variable.

In the discrete attribute map definition, the NAME= option assigns the name of the attribute definition. A VALUE statement is defined for each possible value of the discrete variable. The DISCRETEATTRVAR statement creates a link between the discrete map definition and the discrete variable. The ATTRVAR= option specifies the name to associate between the attribute map and the input column. The VAR= option points to the input column that you are associating with the map and the ATTRMAP= option points to the name of the discrete attribute map to use in the program. Colors can be expressed as color names, RGB values, or HLS values. Figures 9 and 10 show the output that is produced by the following code.

```
proc template;
   define statgraph attributes;
   begingraph;
       entrytitle 'Students';
       discreteattrmap name='colors' / ignorecase=true;
          value 'Cindy' / fillattrs=(color=orange );
          value 'Susan' / fillattrs=(color=cx8c7962);
          value 'Bob' / fillattrs=(color=vlib );
             value 'John' / fillattrs=(color=H0DDbf99);
       enddiscreteattrmap;
       discreteattrvar attrvar=classfill var=student attrmap='colors';
       layout overlay ;
          barchart x=student y=score / dataskin=sheen group=classfill ;
      endlayout;
   endgraph;
  end;
run;

data school;
   input Class $ Student $ Score;
   datalines;
English Bob 65
English Cindy 80
English Susan 30
English John 90
Math Bob 76
Math John 82
Math Susan 90
;
run;

proc sgrender data=school template=attributes;
   by class;
run;
```

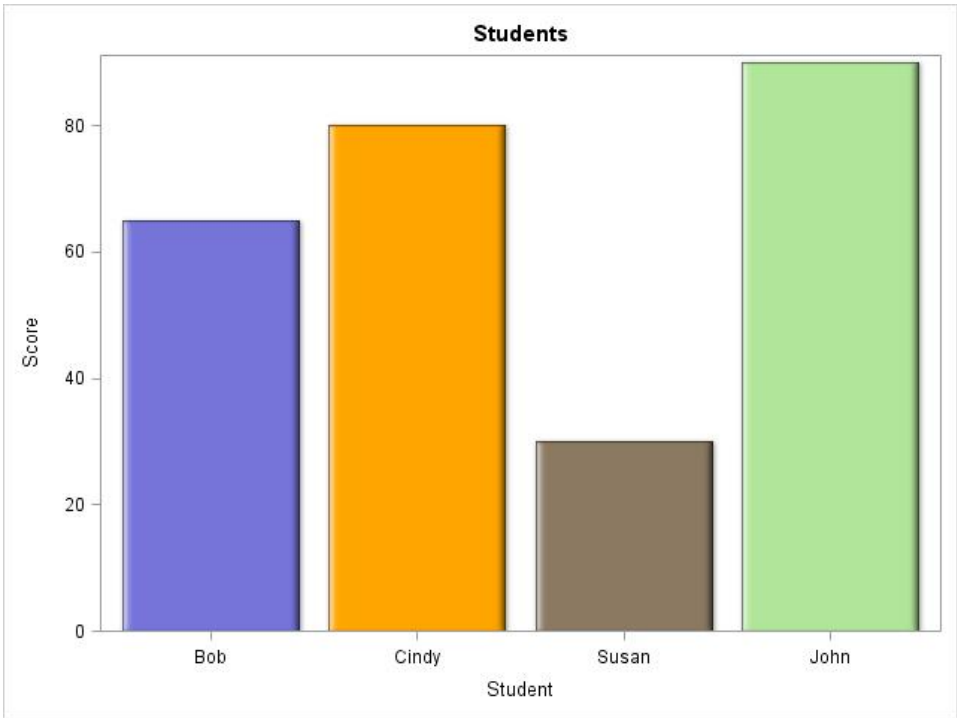

Figure 9. Colors Assigned with a Discrete Attribute Map

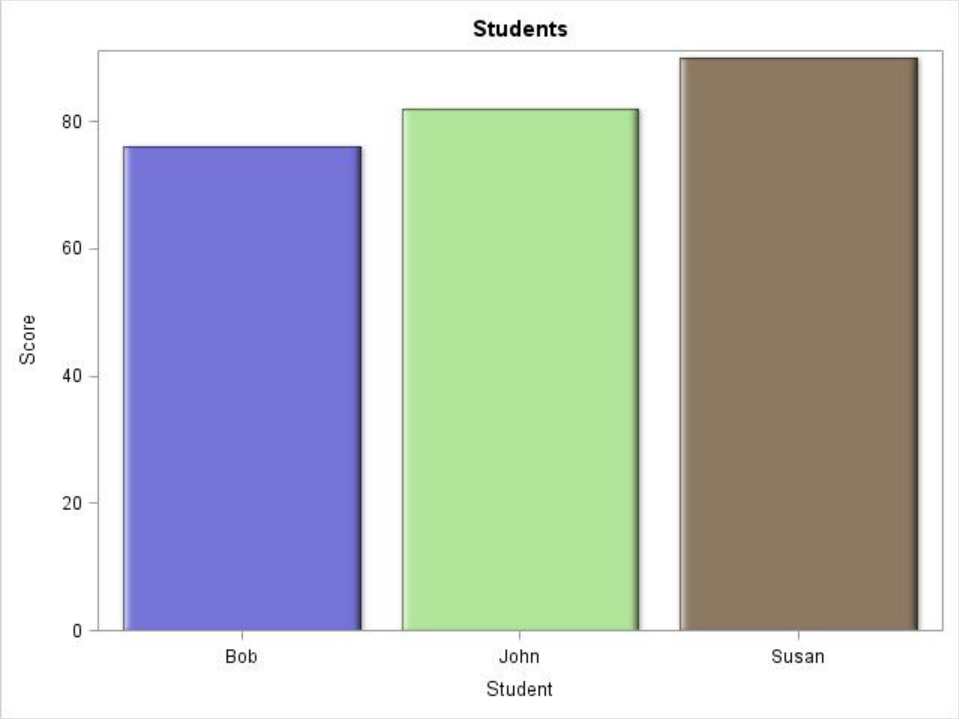

Figure 10. Colors Maintained across Graphs with a Discrete Attribute Map

ADDING A TABLE UNDER A GRAPH

Adding a table under a graph can be cumbersome when using traditional SAS/GRAPH procedures, and it is not possible with PROC SGPLOT and PROC SGPANEL. Luckily, creating a graph with a table is now easier with GTL and the BLOCKPLOT statement. The following example shows you how to define a cell for your graph and a separate cell for the table within a LAYOUT LATTICE block. The ROWWEIGHTS= option in the LAYOUT LATTICE statement defines the size of the cells. The GROUP= option in the BARCHART statement is assigned the midpoint variable so that each bar is a different color. Specifying DISPLAY=(*LABEL*) on the XAXISOPTS parameter in the second LAYOUT OVERLAY statement suppresses the X axis values and tick marks on the BlockPlot. The output that is produced from the following code is shown in Figure 11.

```
proc template;
   define statgraph barblock;
   begingraph;
   entrytitle 'Total Sales Across Regions';
      layout lattice /rowweights=(.65 .35);
      layout overlay;
         barchart x=region y=sales /group=region barlabel=true dataskin=gloss;
      endlayout;
      layout overlay /xaxisopts=
                     (type=discrete label='Number of Stores per Region'
                      display=(label)) walldisplay=none;
         blockplot x=region block=stores/class=product display=
                                          (outline values label) repeatedvalues=true;
      endlayout;
      endlayout;
   endgraph;
   end;

proc sort data=sashelp.shoes out=new;
   by region product;
run;

proc means data=new noprint;
   where region in('Africa' 'Asia' 'Canada' 'Pacific' 'United States');
   id sales;
   by region product;
   var stores;
   output out=shoes sum=;
run;

proc sgrender data=shoes template=barblock;
run;
```

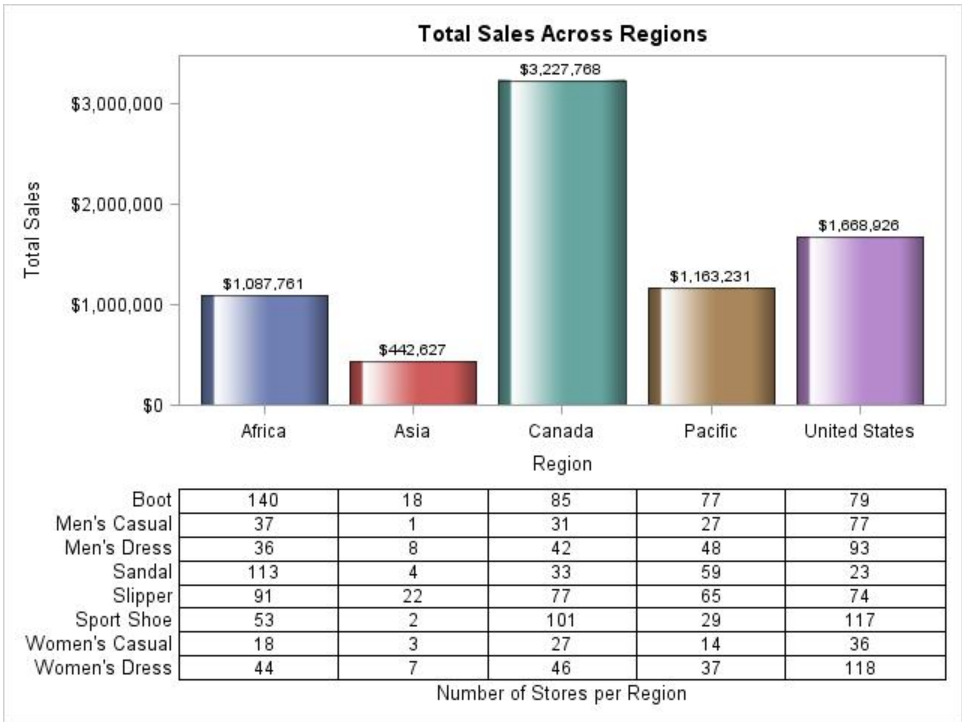

Figure 11. Displaying a Table under a Graph

CREATING A BREAK IN THE AXIS

You might have instances in which there is an extreme difference between some of your response values. In the following example, the response values for midpoints E and F are much higher than the response values for the other midpoints. The output is shown in Figure 12.

```
data new;
   input Type $1 Value ;
   datalines;
A 10
B 15
C 12
D 17
E 205
F 220
;
run;

proc sgplot data=new;
   vbar type / response=value;
   title 'VBAR Chart Created with PROC SGPLOT';
run;
```

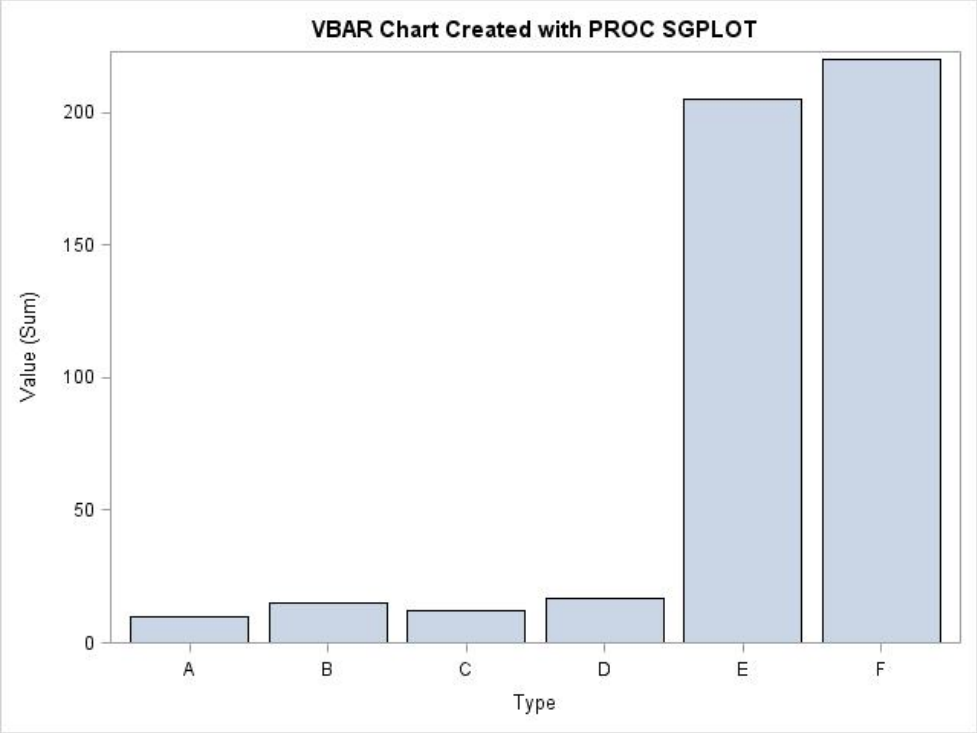

Figure 12. A Vertical Bar Chart with a Wide Range of Values

With GTL, you can create a break in the vertical axis in order to have a better view of the range of data for the bars with the lower values. An easy way to do this is by plotting the lower Y values in the bottom cell and the upper Y values in the top cell. This is accomplished by using GTL to place the two BARCHART statements in separate LAYOUT OVERLAY blocks within a LAYOUT LATTICE block. The output is shown in Figure 13.

```
data new;
   input Type $1 Value ;
   datalines;
A 10
B 15
C 12
D 17
E 205
F 220
;
run;

data split;
   set new;
   if value >200 then value2=value;
      else value2=0;
   output;
run;

proc template;
   define statgraph broken;
   begingraph;
      entrytitle 'Panel Break';
      layout lattice / rows=2 rowweights=(.3 .7) rowdatarange=union rowgutter=9;
         layout overlay / xaxisopts=(display=none)
                          yaxisopts=(display=(tickvalues ticks)
                 linearopts=(viewmin=200 viewmax=220
                             tickvaluesequence=(start=200 end=220 increment=5)));
```

```
            barchart x=type y=value2 / dataskin=matte;
         endlayout;
         layout overlay / yaxisopts=(offsetmax=0 linearopts=(viewmin=0
                          viewmax=25 tickvaluesequence=(start=0
                                                        end=25 increment=5)));
            barchart x=type y=value / dataskin=matte;
         endlayout;
      endlayout;
   endgraph;
end;

proc sgrender data=split template=broken;
run;
```

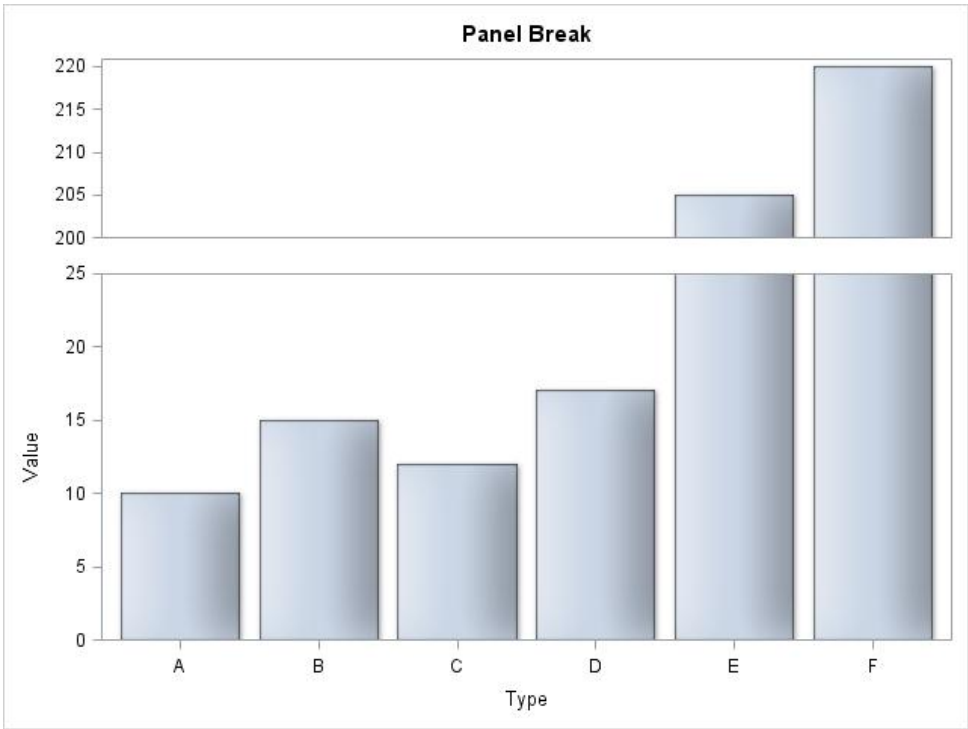

Figure 13. Creating a Break in the Axis to Better Display Data

BUILDING SIDE-BY-SIDE GRAPHS WITH Y AND Y2 AXES

The following example uses GTL to build side-by-side panels with Y and Y2 axes. The YAXIS=Y2 is specified in the second SERIESPLOT statement, producing the right-hand axis. In this example, the HEADERLABELATTRS= option in the LAYOUT DATAPANEL statement specifies the style attributes to be used for the header labels.

```
proc template;
   define statgraph plot;
   begingraph;
   EntryTitle "SERIESPLOT Statement with Y2AXIS" ;
   layout datapanel classvars=(sex) / columns=2 rows=1
                    headerlabelattrs=(color=red weight=bold);
   layout prototype;
      SeriesPlot X=age Y=height / group=dose display=all name='height'
         lineattrs=(color=cx5DAF5D)
            markerattrs=(symbol=squarefilled color=cx5DAF5D);
      SeriesPlot X=age Y=weight / group=dose yaxis=y2 display=all name='weight'
            lineattrs=(color=cx4B50AA pattern=2)
            markerattrs=(symbol=circlefilled color=cx4B50AA);
```

```
   endlayout;
   sidebar / align=bottom;
      discretelegend 'height' 'weight' / ;
   endsidebar;
   endlayout;
   endgraph;
   end;
run;
proc sort data=sashelp.class out=class;
   by age;
run;

proc sgrender data=class template=plot;
run;
```

The output is shown in Figure 14.

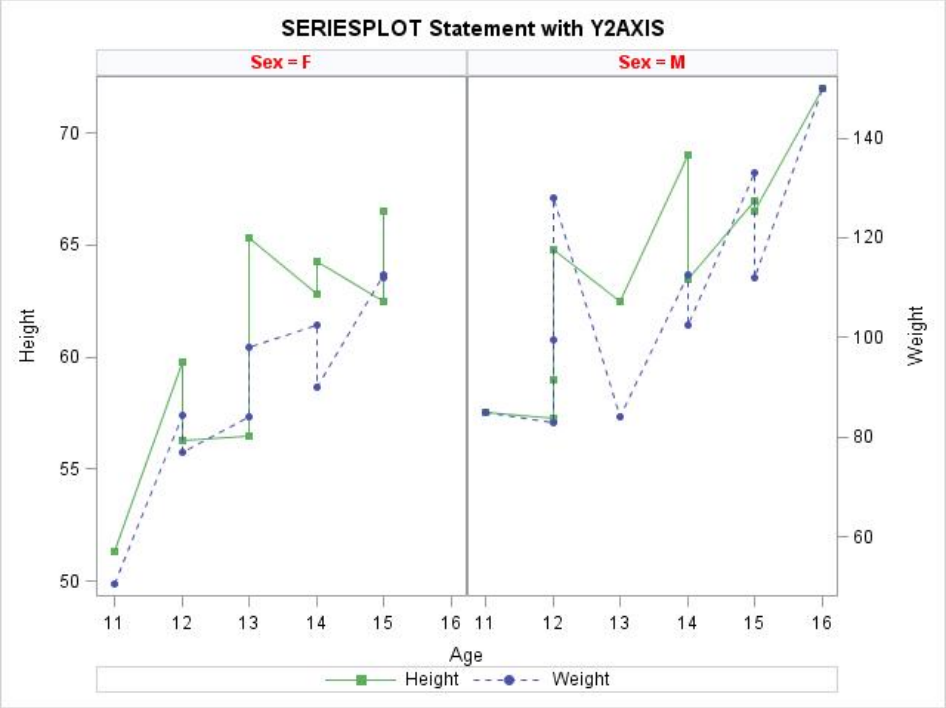

Figure 14. Displaying Two Vertical Axes

Additionally, you can remove the legend and header borders. This is done by defining a style definition with GRAPHBORDERLINES and GRAPHBACKGROUND set to `white` and specifying HEADEROPAQUE=FALSE in the LAYOUT DATAPANEL statement. The STYLE= option in the ODS HTML statement points to the newly defined style definition.

```
proc template;
   define statgraph plot;
   begingraph;
   EntryTitle "SERIESPLOT Statement with Y2AXIS";
   layout datapanel classvars=(sex) / columns=2 rows=1
      headerlabelattrs=(color=red weight=bold) headeropaque=false;
   layout prototype;
      SeriesPlot X=age Y=height / group=dose display=all name='height'
         lineattrs=(color=cx5DAF5D)
         markerattrs=(symbol=squarefilled color=cx5DAF5D);
      SeriesPlot X=age Y=weight / group=dose
```

```
            yaxis=y2 display=all name='weight'
          lineattrs=(color=cx4B50AA pattern=2)
          markerattrs=(symbol=circlefilled color=cx4B50AA);
   endlayout;
      sidebar / align=bottom;
      discretelegend 'height' 'weight' / border=false;
   endsidebar;
   endlayout;
   endgraph;
   end;
   define style noheaderborder;
      parent = styles.default;
         class graphborderlines /
               contrastcolor=white;
         class graphbackground /
               color=white ;
   end;
run;

proc sort data=sashelp.class out=class;
   by age;
run;

ods html style= noheaderborder;
proc sgrender data=class template=plot;
run;
```

The output is shown in Figure 15.

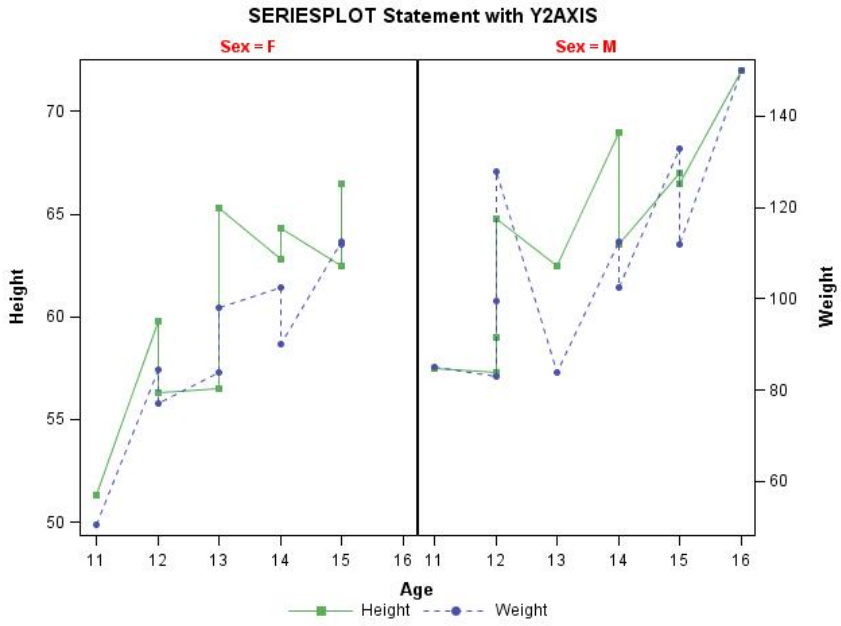

Figure 15. Removing Borders from Headers and the Legend

As discussed in the section "Overlaying a Scatter Plot and Bar Chart," if you overlay a bar chart and a scatter plot in the same cell with PROC SGPLOT, not all plot statements are compatible. This is also the case with PROC SGPANEL. If you want to add error bars on a bar chart, you need to use GTL. With GTL, you can add error bars on a bar chart when using the GROUP= option with the YERRORLOWER= and YERRORUPPER= options in the SCATTERPLOT statement. The output from the following code is shown in Figure 16.

```
proc sort data=sashelp.class out=class;
   by sex;
proc means data=class nway mean clm noprint;
   by sex;
   class age;
   var height;
output out=new mean= lclm= uclm= / autoname;
run;

proc template;
   define statgraph barchart;
      begingraph;
      entrytitle 'Bar Chart with Error Bars';
      layout gridded / border=false;
      layout datalattice columnvar=sex / headerlabeldisplay=value cellwidthmin=50
                    columnheaders=bottom border=false columndatarange=union
                   columnaxisopts=(display=(line tickvalues))
                      rowaxisopts=(offsetmin=0 linearopts=
                                   (viewmax=100 tickvaluepriority=true)
                      label='Mean Height' griddisplay=on);
      layout prototype / walldisplay=(fill);
         barchart x=age y=height_mean / group=sex barlabel=true skin=modern
               name='bar'    outlineattrs=(color=black);
         scatterplot x=age y=height_mean / group=sex yerrorlower=height_lclm
                                                    yerrorupper=height_uclm
                         markerattrs=(size=0) name='scatter'
                         errorbarattrs=(thickness=2) datatransparency=0.6;
      endlayout;
      endlayout;
      endlayout;
      endgraph;
end;
run;

proc sgrender data=new template=barchart;
run;
```

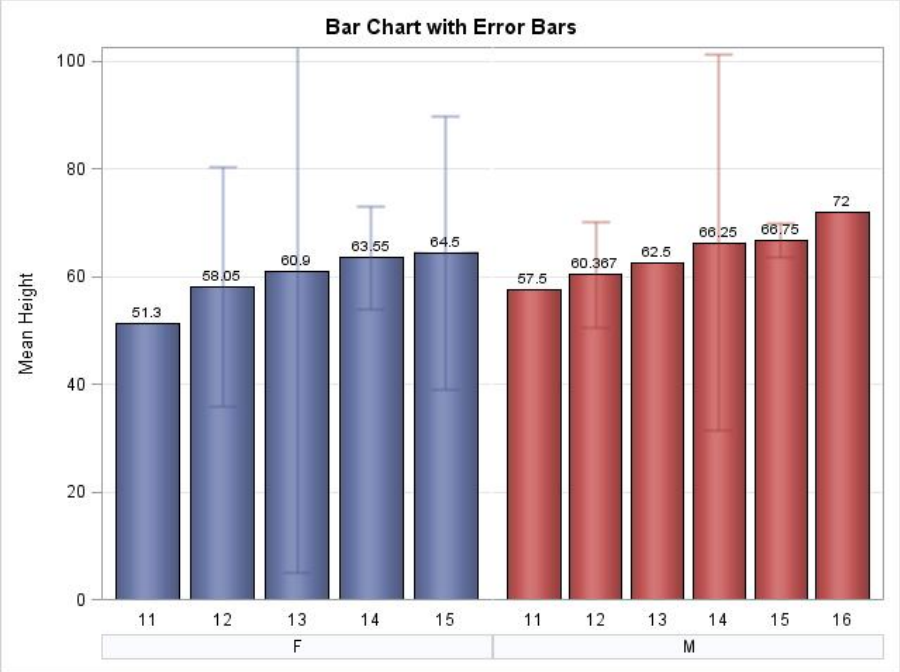

Figure 16. A Bar Chart with Error Bars Displayed in the Same Cell

CONCLUSION

The next time you are unable to produce the graph you want with PROC SGPLOT or SGPANEL, remember that GTL is ready to come to the rescue. With GTL, you can overlay plot types that are not possible with PROC SGPLOT and SGPANEL, split axis values across lines, create graphs with dynamic variables, and much more. The power is right at your fingertips with GTL.

The examples in this paper will be added to the Graphics Samples Output Gallery on the SAS Customer Support site (support.sas.com/sassamples/graphgallery/index.html). As new GTL examples become available, they also will be added to the gallery.

REFERENCES

SAS Institute Inc. 2012. "Special Dynamic Variables" and "Concepts: SGPLOT Procedure." *SAS® 9.3 ODS Graphics Procedures Guide, Third Edition*. Cary, NC: SAS Institute Inc. Available at support.sas.com/documentation/cdl/en/grstatproc/65235/PDF/default/grstatproc.pdf.

CONTACT INFORMATION

Lelia McConnell
SAS Institute Inc.
SAS Campus Drive
Cary NC 27513
E-mail: support@sas.com
Web: support.sas.com