Paper SAS156-2014

# Putting on the Ritz: New Ways to Style Your ODS Graphics to the Max

Dan Heath, SAS Institute, Inc., Cary, NC

## ABSTRACT

Do you find it difficult to dress up your graphs for your reports or presentations? In SAS® 9.4, we have introduced new capabilities in the ODS Graphics system that give you the ability to style your graphs without creating or modifying ODS styles. Some of the new capabilities include the following:

- a new option to control the behavior of how ODS styles are applied
- graph syntax to override ODS style attributes for grouped plots
- enhanced attribute map support
- the ability to define font glyphs and images as plot markers

In this presentation, we will discuss these new features in detail, showing examples in both a Graph Template Language (GTL) context and a statistical graphics (SG) procedure context.

## INTRODUCTION

Styles are an important part of an effective data visualization. Poor style choices can obscure the data and make it difficult to see the message in the graph, while good style choices can enhance the visual and convey the information clearly. There have been a number of papers that have discussed the principles of effective graphics, including "Make a Good Graph" (Matange 2013) and "Creating Presentation-Quality ODS Graphics Output" (Heath 2010). However, this presentation will focus on syntax enhancements to the Graph Template Language (GTL) and the statistical graphics (SG) procedures that will make it easier to style your graphs.

To understand how some of these new features work, it is important to understand how graph style classes work within an ODS style. The part of the style that is important for this discussion involves the following classes:

```
proc template;
    define style styles.mystyle;
    parent=style.listing

    class GraphData1;
    class GraphData2;
    class GraphData3;
    class GraphData4;
    class GraphData5;
    class GraphData6;
    class GraphData7;
    class GraphData8;
    class GraphData9;
    class GraphData10;
    class GraphData11;
    class GraphData12;
    /* More GraphDataN classes can be added */
    end;
run;
```

The GraphData1through GraphDataN classes can not only be referenced directly from syntax like other style classes, but they are also used automatically when a group option (such as GROUP and COLORGROUP) is specified. Within each GraphDataN class, the following attributes are possible:

```
class GraphData1 /
    color = cxFF0000 /* Used for fill colors */
    contrastcolor = cxcc0000 /* Used for line/marker colors */
    linestyle = 1
    markersymbol = "circle"
end;
```

It is not required that GraphDataN classes contain all of these attributes. However, it is required that an attribute be specified contiguously, starting from GraphData1. This means, for example, that I can specify colors for GraphData1 through GraphData12 but specify marker symbols for only GraphData1 through GraphData5. If I were to accidentally specify marker symbols for GraphData1 through GraphData4 and also for GraphData6, the marker symbol in GraphData6 would be ignored because of the "hole" in GraphData5. Only the marker symbols in GraphData1 through GraphData4 would be used. The table in Table 1 makes it easier to visualize how the attributes are mapped to each unique group value.

| Style Class | Color | ContrastColor | LineStyle | MarkerSymbol |
|---|---|---|---|---|
| GraphData1 | Blue | Blue | 1 | Circle |
| GraphData2 | Red | Red | 4 | Plus |
| GraphData3 | Green | Green | 8 | Triangle |
| GraphData4 | Yellow | Yellow | 5 | |
| GraphData5 | Cyan | Cyan | | |

**Table 1. Style Example**

If we were creating a grouped scatter plot, the first group value would be drawn as a blue circle, with all of the attributes coming from GraphData1. The next two unique group values would get all of their attributes from GraphData2 and GraphData3, respectively. However, for the next group value, GraphData4 has a color defined, but not a marker symbol. Therefore, the style processor will loop back up to GraphData1 and use its symbol, giving us a yellow circle. The next value would be a cyan plus, and so on, until all unique group values are mapped. If the colors were exhausted, they, too, would loop back to the beginning. Therefore, it is good to use a prime number of attribute values to help maximize the number of attribute combinations for a large number of group values. For this reason, most of the ODS styles shipped with SAS specify 12 colors, 11 line patterns, and 7 markers.

## ATTRIBUTE PRIORITY

Now that you understand how style attributes are applied to group values, it's time to introduce a new option that can modify that behavior. Using our previous example, the first unique group value would be rendered as a blue circle, the next as a red plus, and so on. But what if you wanted just circle markers for all colors, iterating to the next marker only if you run out of colors? In the past, you would have had to make a custom style that contained enough GraphDataN style classes to embody this behavior (Table 2).

| Style Class | Color | ContrastColor | MarkerSymbol |
|---|---|---|---|
| GraphData1 | Blue | Blue | Circle |
| GraphData2 | Red | Red | Circle |
| GraphData3 | Green | Green | Circle |
| GraphData4 | Yellow | Yellow | Circle |
| GraphData5 | Cyan | Cyan | Circle |
| GraphData6 | Blue | Blue | Plus |
| GraphData7 | Red | Red | Plus |
| GraphData8 | Green | Green | Plus |
| GraphData9 | Yellow | Yellow | Plus |
| GraphData10 | Cyan | Cyan | Plus |
| GraphData11 | Blue | Blue | Triangle |
| GraphData12 | Red | Red | Triangle |
| GraphData13 | Green | Green | Triangle |
| GraphData14 | Yellow | Yellow | Triangle |
| GraphData15 | Cyan | Cyan | Triangle |

**Table 2. Style Expanded on Marker Symbol**

Creating custom styles for these types of situations can be quite cumbersome. However, we now have a new option called ATTRPRIORITY that enables you to set the behavior for group attribute mapping. The default value is NONE, which gives you normal behavior described in the introduction. If you set the ATTRPRIORITY to be COLOR, the color change is given priority when multiple attributes are needed to draw the plot. For example, the line chart in Figure 1 was drawn using the LISTING style with the default attribute priority.

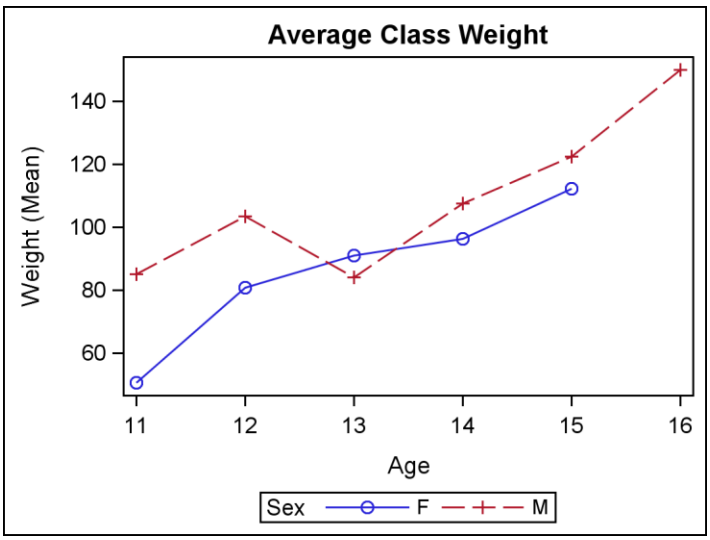


**Figure 1. Line Chart with ATTRPRIORITY=NONE**

```
ods listing;
proc sgplot data=sashelp.class;
   vline age / response=weight stat=mean group=sex markers;
run;
```

For a simple plot like this, you might want the plot lines to stay solid and use the same marker shape. Instead of creating a custom style for that look, you can now just change the attribute priority (Figure 2).

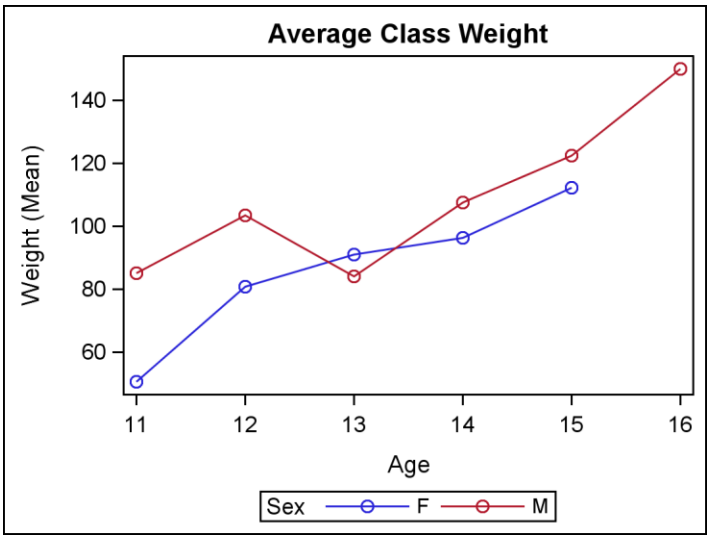


**Figure 2. Line Chart with ATTRPRIORITY=COLOR**

```
ods listing;
ods graphics / attrpriority=color;
proc sgplot data=sashelp.class;
   vline age / response=weight stat=mean group=sex markers;
run;
```

Attribute priority can be specified in any of the following three locations (in order of precedence):

1. in the BEGINGRAPH statement in GTL
2. in the ODS GRAPHICS statement
3. in an ODS style

For graph template creators, there might be certain plot templates that you create where you want to guarantee color priority, regardless of the user's settings or the style settings. In that case, use ATTRPRIORITY=COLOR in the BEGINGRAPH statement (Figure 3). In general, however, the option in BEGINGRAPH should not be used in order to maximize flexibility for users of the template.

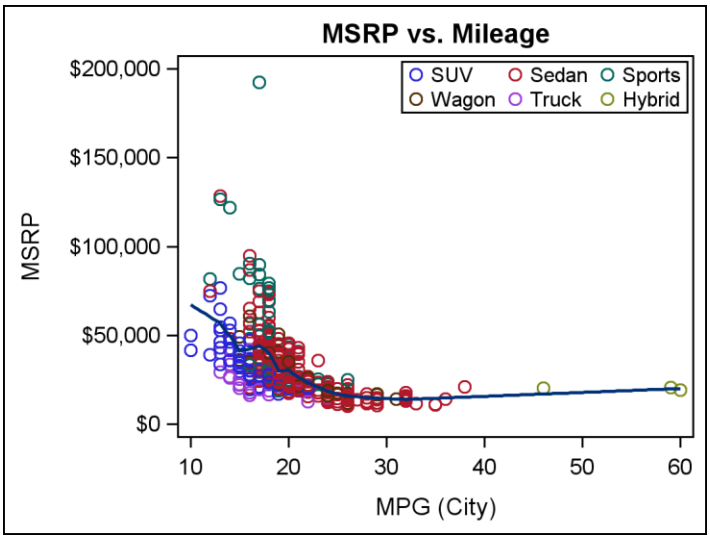


**Figure 3. ATTRPRIORITY=COLOR in the BEGINGRAPH Statement**

```
ods listing;
proc template;
define statgraph mileage;
begingraph / attrpriority=color;
   layout overlay;
      scatterplot x=msrp y=mpg_city / group=type name="scatter";
      loessplot x=msrp y=mpg_city;
      discretelegend "scatter" / location=inside valign=top halign=right;
   endlayout;
endgraph;
end;
run;

proc sgrender data=sashelp.cars template=mileage;
run;
```

Most users will change the priority via the ODS GRAPHICS statement. (See Figure 2.) This option enables you to easily change the priority for different graphs within the same program. In addition, the option enables you to change the priority of any ODS style in the system without having to modify the style.

The location of least precedence for attribute priority is in the ODS style. Using this option in the style enables a style creator to enforce a default priority in the style while still creating the style in a way that supports changing the priority. To set the option in the style, set ATTRPRIORITY in the GRAPH style class. We use this option in our production HTMLBLUE style to enforce color priority. The following code can take your favorite ODS style and turn it into a color priority style.

```
proc template;
define style styles.my_favorite;
parent=styles.listing; /* substitute your favorite style here */
class Graph / attrpriority = "color";
end;
run;
```

Make sure that your parent style is not one of the SAS production black and white or grayscale-based styles, as the color for the markers does not vary.

The ATTRPRIORITY option can also have an effect on non-grouped plots that make direct style references from plot attributes. When ATTRPRIORITY is set to COLOR, the internal style is expanded as described in Table 2. Suppose you had two overlaid scatter plots in the same graph, with MARKERATTRS=GraphData1 in the first plot and MARKERATTRS=GraphData2 in the second. Instead of getting red circles and green squares (Table 1), you will get red circles and green circles (Table 2). Figures 4 and 5 demonstrate this behavior using the LISTING style.

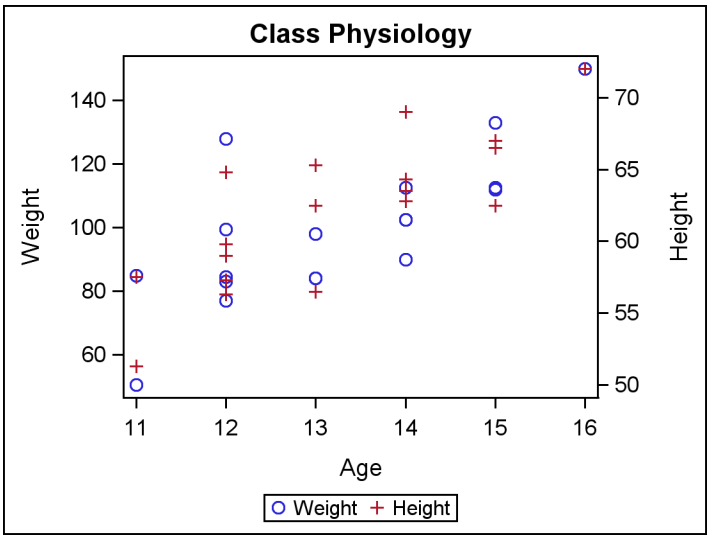


**Figure 4. Scatter Plot with Plot Style Attribute Overrides**

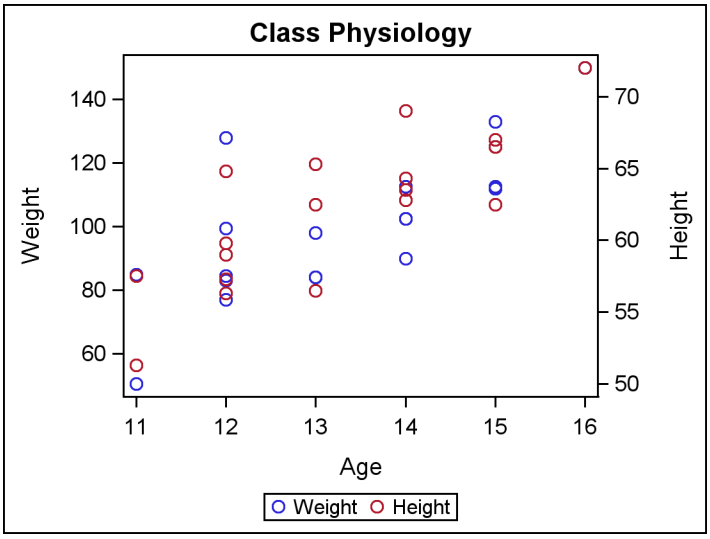


**Figure 5. Scatter Plot with Plot Style Attribute Overrides and ATTRPRIORITY=COLOR**

```
ods listing;
ods graphics / attrpriority=color; /* Use NONE to get Figure 4 */
```

```
Title "Class Physiology";
proc sgplot data=sashelp.class;
   scatter x=age y=weight / markerattrs=GraphData1;
   scatter x=age y=height / markerattrs=GraphData2 y2axis;
run;
```

## OVERRIDING ODS STYLES USING GRAPH SYNTAX

There are many users who are not comfortable with creating or modifying ODS styles, yet they still have the need to control the ODS style attributes used for the graph. For overlaid plots without groups, the attributes for each plot can be modified directly for each plot, as demonstrated in Figures 4 and 5.

Even for grouped plot situations, it is possible to control attributes using the plot syntax rather than modifying the ODS style. Using the example from Figure 1 and Figure 2, if you want to guarantee that your grouped lines are solid while the other attributes vary, you can use the plot syntax to hold the line pattern constant (Figure 6). The danger, however, is that if the number of group values exceed the number of colors, then duplicate plot lines will appear. If there is the possibility of color overrun, it is safer to use the ATTRPRIORITY option instead so that the line pattern will be incremented if the colors are exhausted.

There are now two ways to modify group colors without editing the ODS style. In SAS® 9.3, we introduced new functionality called an *attribute map* that enables you to map visual attributes to specific group values. Now, in SAS 9.4, we have introduced new options to enable you to replace the attributes from GraphData1 through GraphDataN with the values you specify. The following sections describe these two features in detail.

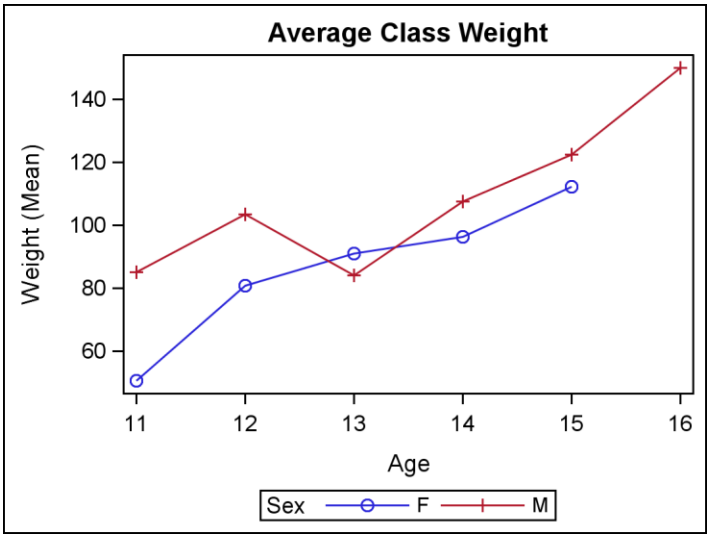


**Figure 6. Line Chart with PATTERN=SOLID**

```
ods listing;
proc sgplot data=sashelp.class;
   vline age / response=weight stat=mean group=sex
               markers lineattrs=(pattern=solid);
run;
```

## REPLACING STYLE ATTRIBUTES

We introduced a new statement in the SG procedures called STYLEATTRS that contains the options to override the GraphData1 through GraphDataN style classes. For GTL users, those options are in the BEGINGRAPH statement. The following options are used to replace the style attributes:

- DATACOLORS—used to replace the fill colors from the GraphDataN classes
- DATACONTRASTCOLORS—used to replace the line and marker colors from the GraphDataN classes
- DATALINEPATTERNS—used to replace the line patterns from the GraphDataN classes
- DATASYMBOLS—used to replace the marker symbols from the GraphDataN classes

Referring back to our example from Table 1, we can use these options to replace an entire attribute column. For example, you could specify DATACOLORS = (cxedf8fb cxb2e2e2 cx66c2a4 cx2ca25f cx006d2c) and DATACONTRASTCOLOR = (Black), and the resulting attribute table would look like Table 3.

| Style Class | Color | ContrastColor | LineStyle | MarkerSymbol |
|---|---|---|---|---|
| GraphData1 | cxedf8fb | Black | 1 | Circle |
| GraphData2 | cxb2e2e2 | | 4 | Plus |
| GraphData3 | cx66c2a4 | | 8 | Triangle |
| GraphData4 | cx2ca25f | | 5 | |
| GraphData5 | cx006d2c | | | |

**Table 3. Line Patterns Replaced by DATALINEPATTERNS**

Notice that the existing colors were removed and replaced by the specified colors. Because these attribute lists are replaced, the effect of direct style references will also change for non-grouped plots, as described earlier with the ATTRPRIORITY option. Figure 7 shows an example of replacing the style colors for both the fill colors and the line colors.

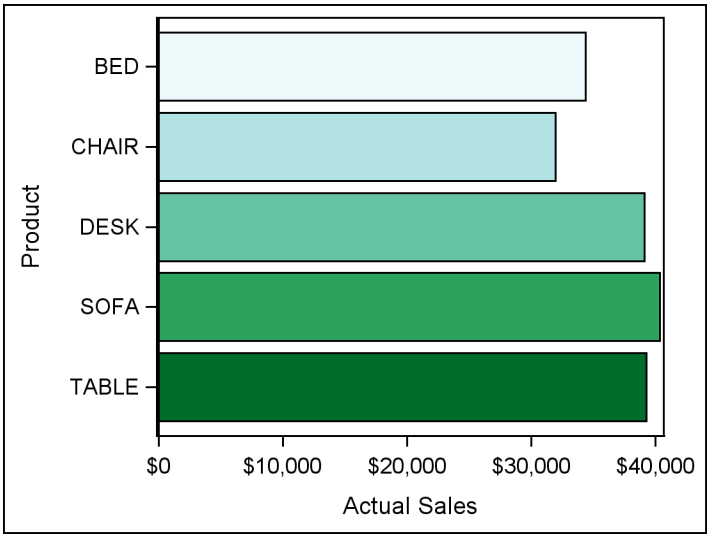


**Figure 7. Bar Chart with New Style Colors**

```
proc sgplot data=sashelp.prdsale noautolegend;
   where year=1993 and quarter<=2;
   styleattrs datacolors=(cxedf8fb cxb2e2e2 cx66c2a4 cx2ca25f cx006d2c)
              datacontrastcolors=(black);
   hbar product / response=actual group=product;
run;
```

The DATACONTRASTCOLORS option was used to force the outline colors to be all black. Otherwise, the existing contrast colors from the style would be used for each bar outline. The code for doing the same chart in GTL would look like the following:

```
proc template;
define statgraph barchart;
begingraph / dataColors=(CXEDF8FB CXCCECE6 CX99D8C9 CX66C2A4 CX2CA25F CX006D2C)
             dataContrastColors=(CX000000);
EntryTitle "1993 sales for Q1 and Q2";
layout overlay;
   BarChart X=PRODUCT Y=ACTUAL / orient=horizontal Group=PRODUCT;
endlayout;
endgraph;
end;
```

```
run;

proc sgrender data=sashelp.prdsale template=barchart;
where year=1993 and quarter<=2;
run;
```

When the style override options are used within the context of an SG procedure, the scope of those options applies only during the run of that procedure. However, when you use these options within a GTL template, the options are used by any procedure that uses that template. For the DATALINEPATTERNS and DATAMARKER options, this usage is usually not much of an issue, but if you specify color overrides, you should use great care. Color overrides should not be used in any GTL templates that are part of a production library in your organization. The reason for this warning is that you do not know what ODS styles will be used with your graph template. Using the wrong ODS style with those color overrides could negatively affect the appearance of your graph, or worse, obscure some of the data. Color overrides in GTL should be made only with templates that are run by people who are aware of the colors in the template.

## ATTRIBUTE MAPS

Although the style override options are good for generally controlling the appearance of grouped plots, there might be times when you need to associate specific style attributes to certain group values. You can try sorting your data to line up with your style attributes, but if some of your group values drop out from your data from one run to the next, then your style attributes will be out of alignment again. In SAS 9.3, we introduced attribute maps to enable you to directly associate group values to attributes, regardless of data order and data drop-out.

Attribute maps can be defined in two ways. For the SG procedures (and now for GTL in SAS 9.4), you can define the map in a SAS data set. For GTL only, you can define the attribute map directly in the template syntax. Currently, the data set version supports only discrete group values (either inherently discrete or made discrete by a SAS format). The GTL syntax supports both discrete attribute maps and range attribute maps (used for continuous variables). Range maps can control only colors, while discrete maps can be used to control the following attributes:

- color
- contrast color
- line patterns
- line thickness (new for SAS 9.4)
- marker symbol
- marker size (new for SAS 9.4)
- fill and marker transparency (new for SAS 9.4)
- text attribute support for AXISTABLE statements (new for 9.4)

Data-based discrete attribute maps used reserved column names to associate group values with attributes. Except for the ID and VALUE columns, all other columns contain attribute values that affect the appearance of the graph. These attribute columns are optional, meaning that you need to define only the columns you need for your graph. However, if you are using attribute maps for many different plot types, it is helpful to go ahead and define the map enough to create a consistent appearance across all of the graphs in your report.

The ID and VALUE columns are required. The ID column is used to identify a particular map in the data set, because you can define multiple attribute maps within the same data set. This ID value is referenced from the procedures that use the map. The VALUE column contains the group values to map to the attributes. It is important to note that the values in the attribute map data set are treated as *case-sensitive*, meaning that the raw data values or the resulting formatted values must have the same case as the map values. Otherwise, the values will not match and the map attributes for that value will not be used.

Figure 8 contains a simple example using a data set attribute map. The light blue and the pink need to be associated with the correct gender to enhance the gender-based plot. The data set was defined with only what was needed for this plot.

```
proc format;
   value agefmt   20-29="20-29"
                  30-39="30-39"
                  40-49="40-49"
                  50-59="50-59"
                  60-69="60-69";
run;

data attrmap;
retain id "gender";
length value $ 6 fillcolor $ 9 linecolor $ 8 makercolor $ 8;
input value $ fillcolor $ linecolor $ markercolor $;

cards;
Male   lightblue blue   blue
Female pink      maroon maroon
run;

Title "Cholesterol Levels by Age Group and Gender";
proc sgplot data=sashelp.heart dattrmap=attrmap;
   format AgeAtStart agefmt.;
   vbox cholesterol / category=AgeAtStart group=sex attrid=gender;
   keylegend / location=inside;
run;
```

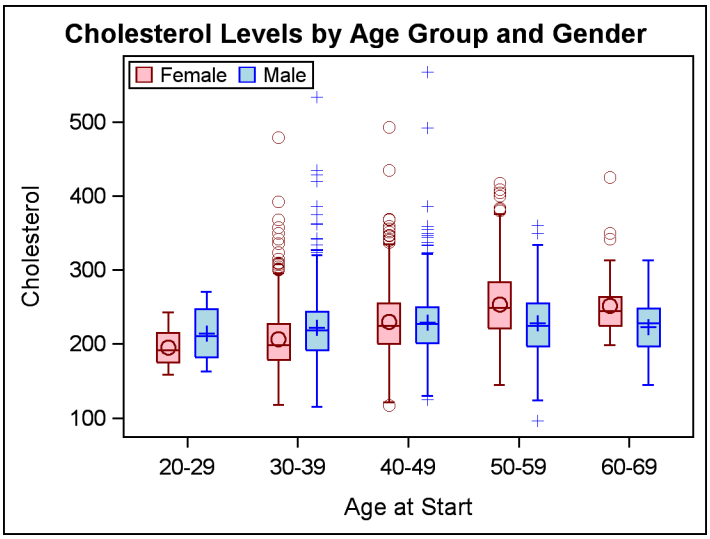


**Figure 8. Box Plot with Discrete Attributes Map**

The attribute map is bound to the SGPLOT output by two options: the DATTRMAP option to reference the data set and the ATTRID option on the plot to reference the desired map within the data set.

In the first maintenance release for SAS 9.4, the ability to specify attribute maps in data sets has been extended to GTL output via PROC SGRENDER. Keeping the same format code and DATA step in Figure 8, the GTL coding equivalent for Figure 8 would be the following:

```
proc template;
define statgraph boxplot;
begingraph;
EntryTitle "Cholesterol Levels by Age Group and Gender" /;
```

```
layout overlay;
   BoxPlot x=AgeAtStart y=Cholesterol / group=sex legendLabel="Cholesterol"
                                        name="VBOX" groupdisplay=cluster;
   DiscreteLegend "VBOX" / Location=Inside autoAlign=(TopLeft TopRight BottomRight
                           BottomLeft Bottom Top Right Left);
endlayout;
endgraph;
end;
run;

proc sgrender data=sashelp.heart template=boxplot dattrmap=attrmap;
   format AgeAtStart agefmt.;
   dattrvar sex="gender";
run;
```

In this case, the binding for the attribute map occurs in the DATTRVAR statement of PROC SGRENDER. The value on the left of the equal sign (=) is the group variable to bind to the map. The quoted value on the right is the attribute map ID from the data set. Multiple assignments can be made in the DATTRVAR statement, or multiple statements can be specified.

Using attribute map data sets instead of embedded GTL attribute map syntax is strongly encouraged, for several reasons:

1. You can create or modify attribute maps without having to recompile the GTL template.
2. As mentioned earlier, you want to avoid embedding colors in your GTL templates used in a library.
3. You can create and reference multiple maps that have colors customized for different ODS styles while using the same GTL template.
4. The data set approach removes any attribute map dependency from the GTL template.

However, if you are using GTL in SAS 9.3, or you need to define a range attribute map for continuous variables, it is important for you to understand how the syntax is constructed. Using Figure 8 again, the GTL template with an embedded discrete attributes map would look like the following:

```
proc template;
define statgraph boxplot;
begingraph;
DiscreteAttrMap name="  ATTRMAP  GENDER";
   Value "Male" / markerattrs=( color=CX0000FF) lineattrs=( color=CX0000FF)
                  fillattrs=( color=CXADD8E6);
   Value "Female" / markerattrs=( color=CX800000) lineattrs=( color=CX800000)
                    fillattrs=( color=CXFFC0CB);
EndDiscreteAttrMap;
DiscreteAttrVar attrvar=GENDER SEX var=SEX attrmap="  ATTRMAP__GENDER";
EntryTitle "Cholesterol Levels by Age Group and Gender" /;
layout overlay;
   BoxPlot X=AgeAtStart Y=Cholesterol / Group=GENDER SEX LegendLabel="Cholesterol"
                                        name="VBOX" groupdisplay=cluster;
   DiscreteLegend "VBOX" / Location=Inside autoAlign=(TopLeft TopRight BottomRight
                           BottomLeft Bottom Top Right Left);
endlayout;
endgraph;
end;

proc sgrender data=sashelp.heart template=boxplot;
   format AgeAtStart agefmt.; run;
```

The DISCRETEATTRMAP block is used to associate formatted discrete values with visual attributes. The attributes are specified using standard GTL syntax for defining fill, line, and marker attributes. The DISCRETEATTRVAR statement is used to bind the attribute map to the group variable. The group variable is specified in the VAR option, and the map name is specified in the ATTRMAP option. The name specified in the ATTRVAR option is then used as the group variable in the plot. If the original variable is used in the plot instead of the ATTRVAR name, the attribute map is completely ignored.

Range attribute maps have a similar construction, but instead of assigning the ATTRVAR name to a discrete group option, you must assign it to a continuous variable option, such as COLORRESPONSE or

MARKERCOLORGRADIENT. In Figure 9, the range map is used to discretely color the bubbles based on the average cholesterol level for the age group. The size of the bubble (and the label) is based on the number of people in each age group.

```
proc template;
define statgraph bubble;
begingraph;
RangeAttrMap name="  ATTRMAP  CHOL";
   Range 100-<200 / rangecolor=green;
   Range 200-<240 / rangecolor=yellow;
   Range 240-400 / rangecolor=red;
EndRangeAttrMap;
RangeAttrVar attrvar=chol range var=cholesterol attrmap="__ATTRMAP__CHOL";
EntryTitle "Average Cholesterol Levels by Age Group";
layout overlay / xaxisopts=(type=discrete);
   BubblePlot X=AgeAtStart Y=Cholesterol size= freq  / datalabel= freq
              datalabelattrs=graphdatatext colorresponse=chol_range name="scat";
   ContinuousLegend "scat";
endlayout;
endgraph;
end;

proc summary data=sashelp.heart nway;
   format AgeAtStart agefmt.;
   class AgeAtStart;
   var cholesterol;
   output out=heart mean=;
run;

proc sgrender data=heart template=bubble;
run;
```

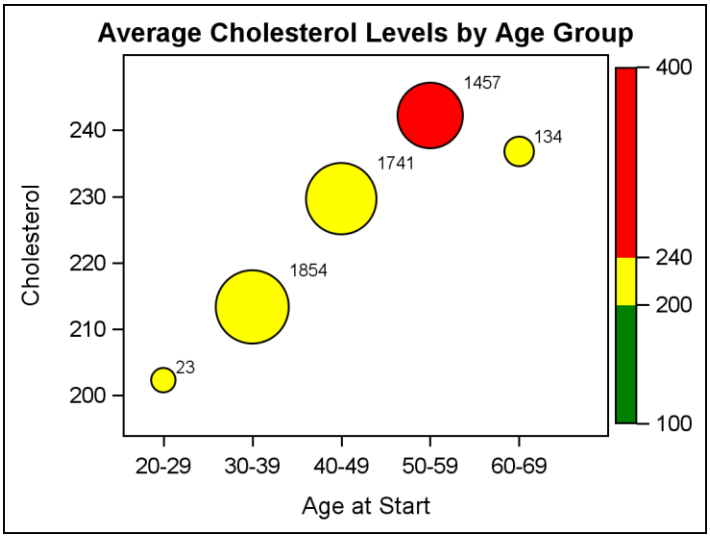


**Figure 9. Bubble Plot with Range Attributes Map**

Typically, you would use the MIN/MAX keywords (or possibly OVERFLOW/UNDERFLOW) to ensure that all of the plotted data was affected by the map. In this case, however, none of the mean values fell into the desirable range. Because of that, using the MIN keyword would not have allowed all of the cholesterol ranges to appear in the legend. Therefore, I used ranges that I knew included all data so that all ranges would appear in the legend.

When you need to create gradient color transitions across ranges, the range attribute map syntax is currently the only way to accomplish this task. However, when mapping continuous ranges to discrete colors (as in Figure 9), it is possible to use a combination of SAS formats and discrete attribute maps to display the plot in a similar way. That

way, you can create the plot with either the SG procedures or GTL, and you can use data-driven maps if desired (Figure 10).

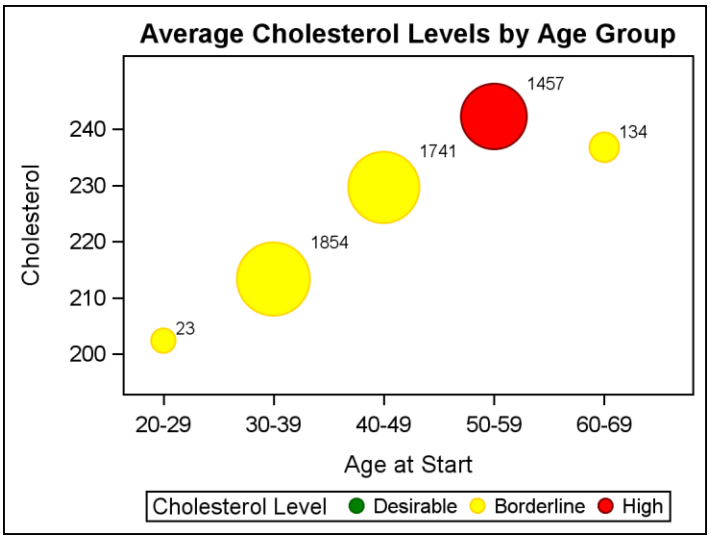


**Figure 10. Bubble Plot with Discrete Attributes Map**

```
proc format;
value cholfmt low-<200 = "Desirable"
              200-<240 = "Borderline"
              240-high = "High";
run;

data attrmap;
retain id "cholrange";
length value $ 10 fillcolor $ 6 linecolor $ 9;
input value $ fillcolor $ linecolor $;
cards;
Desirable  green  darkgreen
Borderline yellow gold
High       red    maroon
;
run;


proc summary data=sashelp.heart nway;
format AgeAtStart agefmt.;
class AgeAtStart;
var cholesterol;
output out=heart mean=;
run;

data heart2 (drop=_temp_);
set heart;
label cholgroup="Cholesterol Level";
if ( n  = 1) then do;
   temp =cholesterol;
  cholgroup = 100;
  cholesterol=.;
  output;
  cholgroup= temp ;
  cholesterol=_temp_;
  output;
end;
else do;
  cholgroup=cholesterol;
```

```
  output;
end;
run;

Title "Average Cholesterol Levels by Age Group";
proc sgplot data=heart2 dattrmap=attrmap;
   format AgeAtStart agefmt. cholgroup cholfmt.;
   xaxis type=discrete;
   bubble x=ageatstart y=cholesterol size= freq  / group=cholgroup datalabel=_freq_
          datalabelattrs=GraphDataText attrid=cholrange;
run;
```

The CHOLFMT format was created to map the cholesterol values to discrete ranges that can be mapped in the ATTRMAP data set. The DATA step for the HEART2 data does two important things for us:

1. It copies the cholesterol values to another column called CHOLGROUP so that we can assign the CHOLFMT format without affecting the axis variable.
2. It prepends a dummy observation with a CHOLGROUP value in the desirable range so that the entry will appear in the legend.

## MARKERS, MARKERS, AND MORE MARKERS

As of SAS 9.4, there are 31 pre-defined markers in the ODS Graphics, including 9 filled markers. Now, with the first maintenance release for SAS 9.4, the possible markers have been greatly extended with the ability to use font glyphs or images as markers.

### USING FONT GLYPHS AS MARKERS

The SGPLOT and SGPANEL procedures, as well as GTL, have a new statement called SYMBOLCHAR that is used to define a new marker name from a font glyph reference. After the symbol is defined, it can be used in any location that takes a marker symbol value, including the new DATASYMBOLS option used to override the ODS style marker symbols. Figure 11 shows an example of defining two markers (male and female) and using them as style overrides for a gender-based variable.

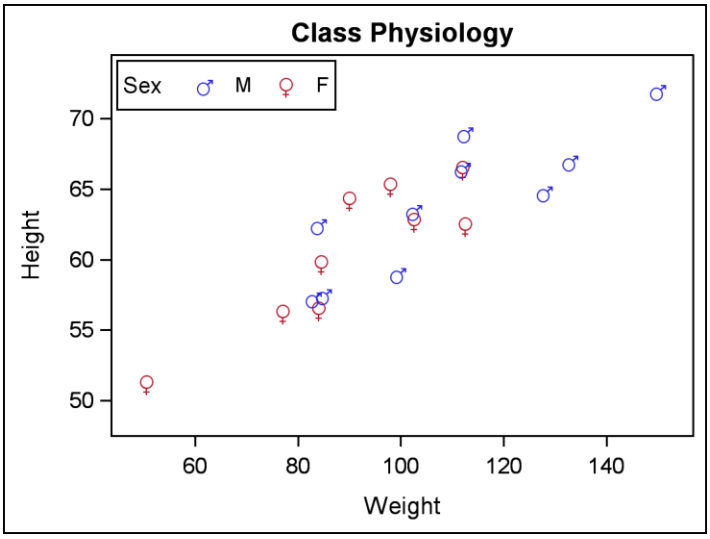


**Figure 11. Scatter Plot with Custom Symbols**

```
title "Class Physiology";
proc sgplot data=sashelp.class;
   symbolchar name=female char='2640'x;
   symbolchar name=male char='2642'x;
   styleattrs datasymbols=(male female);
   scatter x=weight y=height / group=Sex markerattrs=(size=20px);
   keylegend / location=inside title="Sex";
run;
```

The SYMBOLCHAR statement takes either a Unicode character specification or one of our predefined keywords for commonly used Unicode symbols. Optional capabilities include modifying text attributes, scaling, rotating, and offsetting the symbol around the plot location.

## USING IMAGES AS MARKERS

In addition to the SYMBOLCHAR statement, we also have the SYMBOLIMAGE statement for defining a marker from an image file. The required arguments are the new symbol name and the image file. The same scaling, rotating, and offsetting options from SYMBOLCHAR also exist on this statement.

At SAS Global Forum 2011, I presented a paper called "Now You Can Annotate Your Statistical Graphics Procedure Graphs" (Heath 2011). On pages 11 and 12, I showed examples of how you could use annotation to draw images as scatter points and curve labels on a series. At the time, annotation was the only way you could use images as points. Now, no annotation is required! Figure 12 shows a similar example using the SYMBOLIMAGE statement.

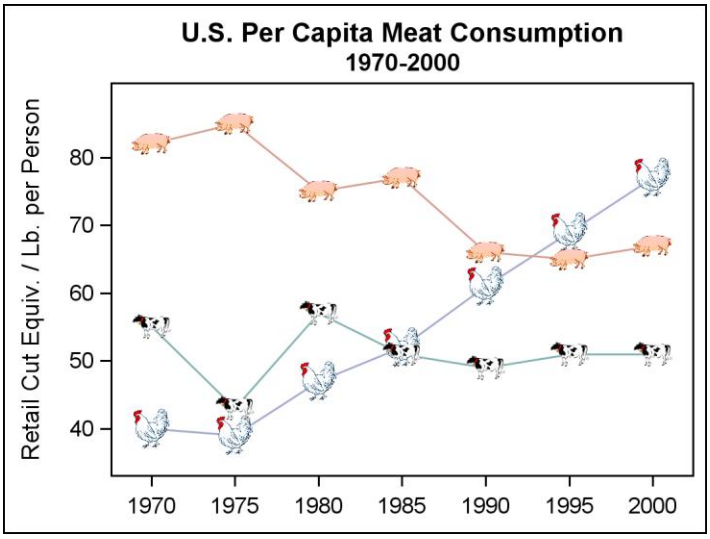


**Figure 12. VLINE Plot with Image Symbols**

```
data meat consumption;
informat year monyy7.;
format year year.;
input year chicken beef pork;
cards;
jan1970 40  82  55
jan1975 39  85  43
jan1980 47  75  57
jan1985 52  77  51
jan1990 61  66  49
jan1995 69  65  51
jan2000 77  67  51
;
run;

Title1 "U.S. Per Capita Meat Consumption";
title2 "1970-2000";

proc sgplot data=meat consumption noautolegend;
xaxis display=(nolabel);
yaxis label="Retail Cut Equiv. / Lb. per Person";
symbolimage name=chicken image = "C:\chicken.png";
symbolimage name=cow image = "C:\cow.png";
symbolimage name=pig image = "C:\pig.png";
series x=year y=chicken / markers markerattrs=(symbol=chicken size=23px)
       lineattrs=(color=CXa5aace pattern=solid);
series x=year y=pork / markers markerattrs=(symbol=cow  size=23px)
```

```
        lineattrs=(color=CX84b2b3 pattern=solid);
  series x=year y=beef / markers markerattrs=(symbol=pig  size=23px)
        lineattrs=(color=CXd69e94 pattern=solid);
  run;
```

When using images as markers, there are a couple of important considerations. First, try to use an image that is approximately the size of the desired plot point size. That way, you will get the best quality rendering of the image. Second, use transparency in whitespace (non-important) parts of the image. For example, if you used a circular company logo, you would want the four corner areas to be transparent. The reason for this is so that, when you use this marker image in a series plot, the line will appear to go all the way to the logo. In Figure 12, notice how the line goes all the way up to the animals' bodies. If the whitespace area was not transparent, the line would have appeared to be clipped. Having transparent whitespace is also important for standard scatter plot applications, because whitespace on one scatter point could cause unnecessary clipping on an adjacent point.

## CONCLUSION

We have discussed a few different ways to override the ODS style for grouped plots without having to modify the actual style. The ATTRPRIORITY option helps you address a very common case where you want the plot lines to be solid across all group values, while still having the flexibility to change to another line pattern if the colors are exhausted. The STYLEATTRS statement and the style attribute override options are great for changing the overall look of a grouped plot when you do not care how the attributes are assigned to the group values. One other thing to consider before using these override options: if you are creating a report with a large number of graphs, and you want all of the graphs to have the same styling, you might want to consider deriving a new style from an existing ODS style to store your settings. Doing this could make it easier to create and maintain your program. Finally, we discussed how the number of available plot marker shapes has been greatly extended by the new support for using font glyphs and images. Hopefully, you will find these new features useful in your own work.

## REFERENCES


- Heath, Dan. 2010. "Creating Presentation-Quality ODS Graphics Output." *Proceedings of the SAS Global Forum 2010 Conference*. Cary, NC: SAS Institute, Inc. Available at support.sas.com/resources/papers/proceedings10/TOC.html.
- Heath, Dan. 2011. "Now You Can Annotate Your Statistical Graphics Procedure Graphs." *Proceedings of the SAS Global Forum 2011 Conference*. Cary, NC: SAS Institute, Inc. Available at support.sas.com/resources/papers/proceedings11/TOC.html.
- Matange, Sanjay. 2013. "Make a Good Graph." *Proceedings of the SAS Global Forum 2013 Conference*. Cary, NC: SAS Institute, Inc. Available at support.sas.com/resources/papers/proceedings13.


## ACKNOWLEDGMENTS


Special thanks to Cynthia Brewer and her excellent site colorbrewer2.org for the colors used in Figure 7.


## RECOMMENDED READING

- *Graphically Speaking Blog* (blogs.sas.com/content/graphicallyspeaking/)

## CONTACT INFORMATION

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


Dan Heath
SAS Institute, Inc.
SAS Campus Dr.
Cary, NC 27513
(919) 677-8000
Dan.Heath@sas.com