Paper 3080-2015

Picture-Perfect Graphing with Graph Template Language

Julie VanBuskirk, MPH, Nurtur Health, Inc., Dallas, TX
Baylor Scott & White Health Care System, Dallas, TX

ABSTRACT

Do you have reports based on SAS®/GRAPH procedures, customized with multiple GOPTIONS? Do you dream of those same graphs existing in a GOPTIONS and ANNOTATE free world? Recreating complex graphs using Statistical Graphics (SG) procedures is not only possible, but much easier than you think! Using before and after examples, I will discuss how the graphs were created using the combination of Graph Template Language (GTL) and the SG procedures. This method produces graphs that are nearly indistinguishable from the original. This method simplifies the code required to make complex graphs, allows for maximum re-usability for graphics code, and allows changes to cascade to multiple reports simultaneously.

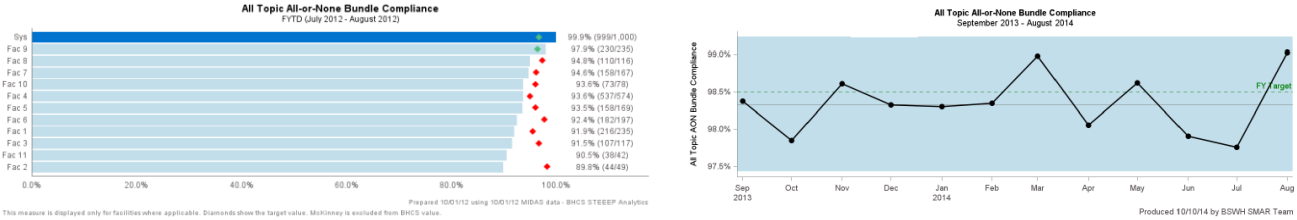
INTRODUCTION

Generally conversion of existing and working code to new SAS® procedures is not a high priority. However, due to the number of reports being produced on a monthly basis, and the requirement that reports all follow a similar marketing approved style definition my department made the decision to invest time into updating to SG procedures and creating a consistent template for graph production. Over time this has proved to be a great investment. The benefits of using graph templates include re-usability, consistency of results, ease of maintenance and updating.

To get the maximum benefit from these methods, I created a production style library. This contains the corporate customized style template along with all graph templates. Thanks to our SAS Administrator, this library is automatically assigned for all programmers when opening Enterprise Guide. This means all programmers automatically get the latest version of the templates and all reports created are consistent. This does require consistency in creating the data sets to be graphed, but with proper documentation it is not difficult for other programmers to adopt these practices. The biggest time savings achieved using this method so far has resulted from a merger that caused total branding changes of our company. The re-branding required a complete overhaul of all reports, there were new colors and fonts which normally would have required manual change in every report. However, since this process had already been adopted the changes in the style and graph templates and automatically cascaded to all production reports.

This paper will specifically focus on creating the complex and custom styled graphs using PROC TEMPLATE. It will cover how the graphs layer when built and how to get the visualization of annotation without actually using ANNOTATE. We will focus on creating the horizontal bar charts and control charts shown in figure 1.

Figure 1.



SETTING UP THE GRAPH TEMPLATE

Some elements of the graph template are consistently used throughout all of the templates and assist in setting up the graph attributes. We will begin by dissecting these pieces, mainly focusing on the dynamic elements and core style updates.

```
Proc template;
```
Initiate template creation

```
    define statgraph b_style.bayor_hbar_large;
```
Define style name. Note, I included a library to make this a permanent style.

The dynamic statement is used to create all reusable elements needed in the template, these are to be used instead of macro calls to make your templates more flexible. Anything from a variable name, text statement, style attribute, etc. can be assigned to a dynamic. *Do not use a macro call in your template, it can cause unexpected results* These dynamics can be named almost anything, however be careful not to name them the same as a variable being used by the template.

```
        dynamic TITLE1 TITLE2 FOOT1 FOOT2 FOOT2 FOOT4
                _LOCATION_CODE2 _DISPLAY _MAX
                 _LOCATION_CODE _METRIC_VALUE
                  _TARGETVAR1 _TRAFFIC _LABEL;
        begingraph/backgroundcolor=CXFFFFFF border = false;
```
Define all dynamic elements. Anything that needs to be easily changed between runs should be defined here.

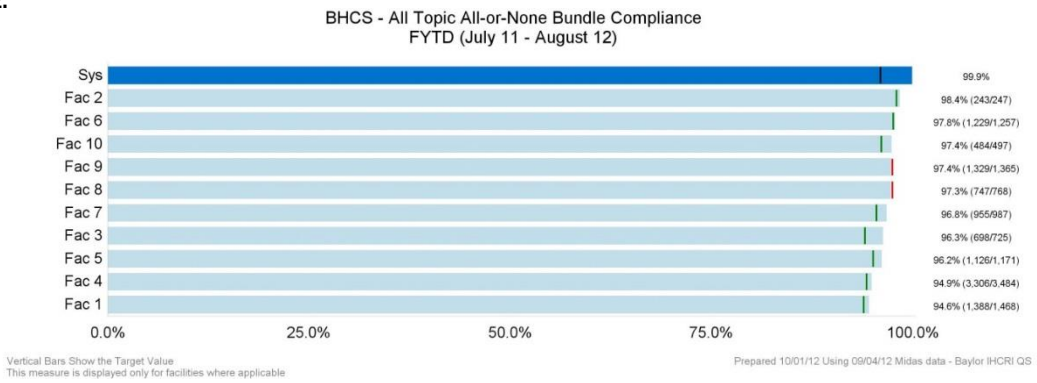
Begin actual graph, set background to white and remove border around the image.

Titles and footers are set following the beginning of the graph statement. Note that the text is replaced with dynamic calls, this allows all users to easily set their own text when running the graphs. The style attributes shown are set to define the font used, color, size, and weight of the text. Also, the order in which the entry statements appears determines the order of the titles and footers. Title1 will always appear at the top and Title2 will be directly below it. We will cover defining the dynamics when we discuss running the graphs using SGRENDER. These hard coded style attributes can be replaced by creating a style template, this is beyond the scope of this paper.

```
        entrytitle TITLE1
                /textattrs=(font='Arial' color=black size =12pt weight = bold);
        entrytitle TITLE2/textattrs=(font='Arial' color=black size =12pt);
        entryfootnote halign =right FOOT1
                        /textatter=(font='Arial' color = dargr size=9pt);
        entryfootnote halign =left FOOT2 FOOT3 FOOT4
                        /textatter=(font='Arial' color = dargr size=9pt);
        layout ….
        /*Define graph elements*/
        endlayout;
    end;
run;
```
Use dynamic for changing titles and footnotes throughout. The styles are then set specific for each. Titles are default centered, alignment can be changed using halign.

HORIZONTAL BAR CHARTS

Figure 2.



This process was based on re-creating an existing graph. This means that I knew in advance what the final product should look like and just needed to figure out how to create a similar style using a template. The image in figure 2 is a representation of the final style that was required. This graph is a jpeg image created using PROC GCHART and GOPTIONS to control the color, font type, and sizing. ANNOTATE was used to add the red and green target lines and the data labels on the horizontal bars. The titles, footers, x-axis, and y-axis were all styled by using in-line style options hard coded into the graphics code.

Graph templates can make more sense if they are thought of as creating an image using a series of layers. Similar to an old fashioned overhead projector with transparencies, multiple transparencies can be overlaid to create an entire picture one step at a time. The image from the first transparency can be seen through the subsequent layers. Also, the order in which the layers are created absolutely matters. If the graphs are put in the wrong order then some elements can be covered by others.

Attribute Maps:

The attribute map is not required for the horizontal bar chart statement, so we will discuss it first. These maps create a data driven approach to coloring the bars. This can be done either as a hard coded set of statements or by using a table. This approach uses the table below, each of the variables shown are required. The attrvar maps to the group statement in the barchart statement, this just provides a name for the association between the attribute map and the data column being matched. The var statement defines which variable to map to and the attrmap specifies a name for the map, this value must match the ID value in the attribute map data set.

```
discreteattrvar attrvar=unitcolor var=_LOCATION_CODE2 attrmap="location_code" ;
```

	ID	VALUE	LINECOLOR	FILLCOLOR
1	location_code	BHCS	CX008FBE	CX008FBE
2	location_code	BSWH	CX008FBE	CX008FBE
3	location_code	BSWH-NTX	CX008FBE	CX008FBE
4	location_code	BASMC	CXC2DEEA	CXC2DEEA
5	location_code	BSW	CXC2DEEA	CXC2DEEA
6	location_code	BUMC	CXC2DEEA	CXC2DEEA

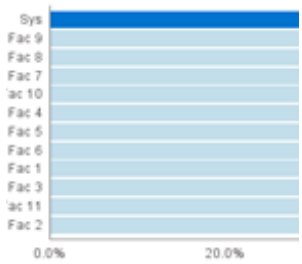
Create Graph Layout:

```
layout lattice;
  layout overlay /
```

Create outer layout, using this could enable you to add multiple overlays
Contains the content for the graph cell – statements for the bar chart

Define X and Y axis options:

The x and y axis options are set immediately after the layout overlay statement and define the graph area. These statement define the format of the tick values (tickvalueformat), the style of the tick values (tickvalueattrs), the maximum and mimimum values shown on the axis (viewmin/viewmax), which elements are displayed (display=) and the offset of the axis (offsetmin/offsetmax – this will be discussed later).




```
xaxisopts=(linearopts=(tickvalueformat=_DISPLAY    viewmin = 0 viewmax = _MAX)
                       display=(tickvalues line ticks)
                       tickvalueattrs = (font= 'Arial' color= dargr size=8pt)
                       offsetmin = 0   offsetmax =  0.8 )
yaxisopts=( display=(line tickvalues )
                      tickvalueattrs =(font= 'Arial' color= dargr size=9pt))
```

The First Graph:

The first graph to be defined should be the one that is the most solidly colored and has the potential to cover up other graph elements. In this case, the piece that takes up the most area is the horizontal bar chart. This was defined as a barchartparm – this requires pre-summarized data, it will produce unexpected results if there is more than one observation for each y-value. The x and y variables are set using dynamics, the group value matches to the attribute map, and the orientation is set via orient option.

```
barchartparm x=_LOCATION_CODE y=_METRIC_VALUE  / group=  unitcolor
                                                 name='bar(h)' orient=horizontal;
```

The Second Graph:

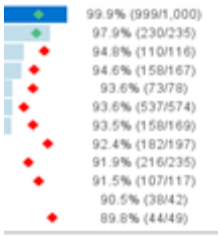
Now the bar chart will be overlaid to get the necessary colored target values. In the original graph, these were shown as small vertical lines, this was updated to diamond (markersymbol) targets for increased visibility. This statement comes second and therefore will always cause the diamond to overlay the bar. Note that the barchartparm x value has to be the y value for this scatterplot, switching the orientation forces the swap in the scatterplots. A single variable is used to define the values of the diamond targets, the color is defined by a secondary flag variable. This maps the variable (defined in markercolorgradient) to the correct color model which is defined in the style template with a start, neutral, and end value. The flag variable in this case is set to 1 = green/start and 3 = red/end. The color model can be hard coded to avoid creating a custom style template.

```
scatterplot x=_TARGETVAR1 y=_LOCATION_CODE / markerattrs=(markersize = 9px
                                             markersymbol = "diamondfilled")
                                             markercolorgradient= traffic_flag
                                             colormodel=threecolorramp;
```

The Final Graph:

The final piece is adding the the labels on the horizontal bars. Setting these as bar labels would cause interference with the target diamonds, especially since there is no way to adequately predict where the target will fall in relation to the bar. So to create the needed look it requires separating the single graph into two distinct areas and then creating a scatterplot to place the label text.

The x axis needs to be split to ensure that the target diamonds and the bar labels do not collide. A second x axis with no display is added. The offset min and offset max are set in both the original x axis and this x2 axis, in this case the primary x takes up 80% of the area and the x2 takes the remaining 20%. I generally group this statement with the other axis statements.




```
x2axisopts= (display= none
             tickvalueattrs = (font= 'Arial' color= dargr size=9pt)
             offsetmin= 0.2 offsetmax = 0  ) ;
```

Now the scatterplot statement is added. This is defined as using the x2 axis. Instead of setting the scatter markers as shapes these are set to a text variable (markercharacter) that contains the specific value(numerator/denominator) that needs to be printed. To change the style attributes of this type of marker requires the use of markercharacterattrs.

```
scatterplot x=xlabel y=_LOCATION_CODE  / markercharacter= _DATA_LABEL
                                         markerattrs= (font= 'Arial' color= dargr size=9pt)
                                         markercharacterattrs=(font= 'Arial' color= dargr size=9pt)
                                               xaxis=x2  ;
```

```
endlayout;
endlayout;
```

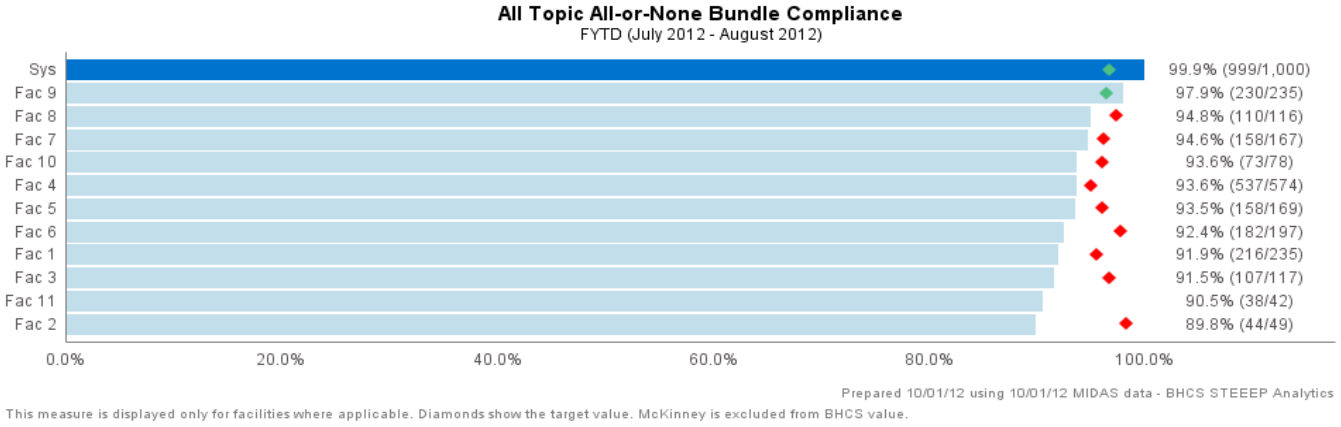
End the overlay layout. A second overlay can follow this statement.

End the lattice layout. This ends the entire graph grid.

The Final Product:

The final product is shown below in Figure 3. A single horizontal bar statement overlaid with multiple scatters to achieve the same basic image as the SAS/GRAPH statements previously used. All of this is done without annotate and in a reusable and potentially data driven method.

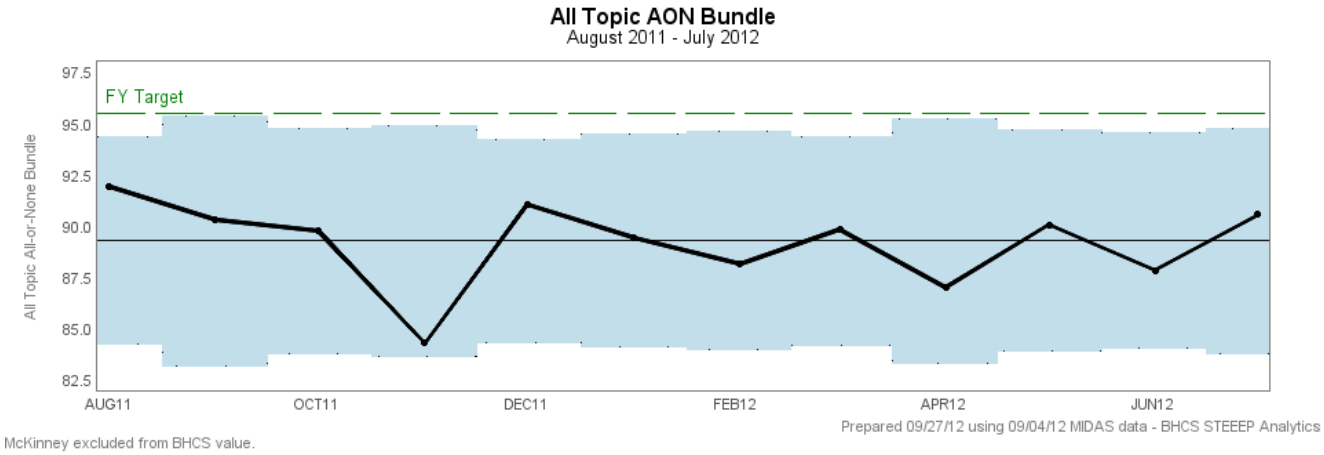
Figure 3.



CONTROL CHARTS

The original control charts were created as output from the PROC SHEWHART procedure which is shown in Figure 4. Creating custom graphs using SHEWHART requires a large number of hard coded style options to be set and requires quite a bit of processing time to create numerous graphs. The SHEWHART procedure is now used to add the required data elements to the reporting dataset, these elements include the upper and lower limits and value for mean line.

Figure 4.



Create Graph Layout:

layout overlay / **Contains the content for the graph cell – statements for the control chart**

Define X and Y axis options:

The x and y axis statements for the control chart are very similar to that of the horizontal bar chart. In this case the label option was added using a dynamic to allow it to be easily changed across metrics. The viewmin and viewmax are also used to force a full axis even if the metric has the same value for an entire year.

```
yaxisopts = (label = _YAXIS   display=(ticks tickvalues label line)
             linearopts=(tickvalueformat=_DISPLAY viewmin = _MIN viewmax = _MAX)
             tickvalueattrs = (font= 'Arial' color= dargr size=9pt )
             labelattrs =(font= 'Arial' color= dargr size=9pt ))
xaxisopts=  (display =( ticks tickvalues line)
            tickvalueattrs = (font= 'Arial' color= dargr size=9pt );
```

The First Graph:

In order to get the same basic image of the control chart several different graphs will have to be layered. To start, we must create the bands with upper and lower limits. The bandplot procedure allows for this type of visualization. Setting the type to step will allow for the potential for the upper and lower limits to change by month, the center justification centers the step on the month, using extend causes the band to fill the entire graph area.

```
bandplot x=rep_period_date limitlower=_lower_ limitupper=_upper_ /  display=(fill) type= step
                           justify=center extend=true fillattrs=(color=CXC2DEEA);
```

The Second Graph:

The next layer requires adding in the mean line printed by default by the SHEWHART procedure. Since this is a single value for all months the reference line procedure can be used. This is simply set to gray.

```
referenceline y= ref1/ lineattrs= (color=gray);
```

The Third Graph:

Next the green target line is added, again this is a single value for all months, so it can be set using the reference line. The curvelabel options are added to create a changeable target label (curvelabel), set it inside the graph area (curvelabellocation), color it green like the line (curvelabelattrs), and control it's position (curvelabelposition).

```
referenceline y= _targetvalue/ lineattrs= (color=green pattern= shortdash)
                curvelabel= _targetlabel curvelabellocation=inside
                curvelabelattrs=(color=green) curvelabelposition=max;
```

The Final Graph:

Finally the actual series plot needs to be added to display the values of the metrics. The options used here set the line to black and define the markers as circles.

```
seriesplot x= _date_var y= metric_value /display=all lineattrs= (color=black)
                markerattrs= (markersize = 9px
                markersymbol = "circlefilled");
```

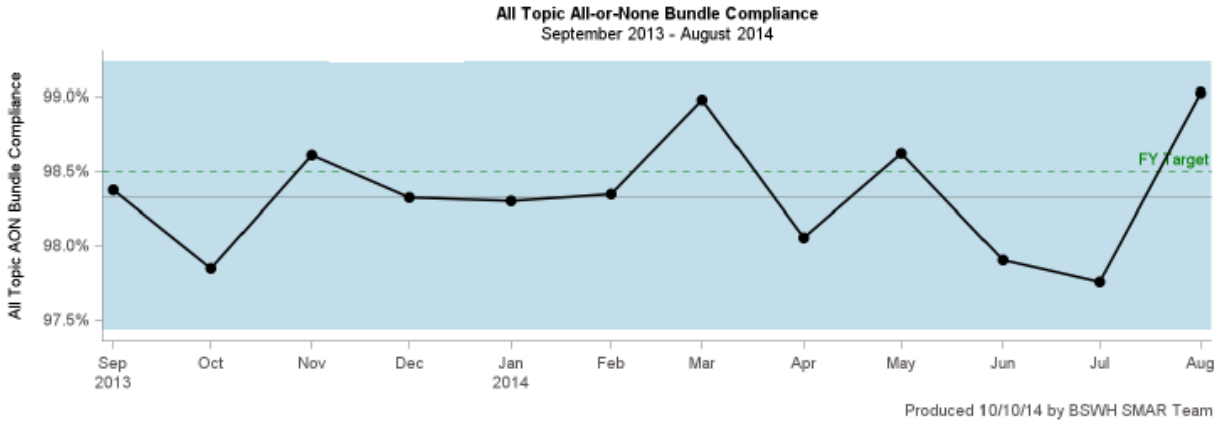
```
endlayout;
```

End the overlay layout. A second overlay can follow this statement.

The Final Product:

The final product is shown below in Figure 5.The combination of the band plot, series plot, and reference lines allow for an improved control chart appearance and can allow the entire graph creation to be data driven and automated.

Figure 5.



ADDITIONAL CONSIDERATIONS:

After the basics of creating complex graphs with GTL are grasped it is fairly simple to make changes and create even more complex results. Adding additional lines just requires updating the GTL to include more layers. If lines are created in the template that have no data to feed them then running the template code will result in a warning that those lines were not produced, however the actual graph will still produce correctly. Starting in SAS 9.4, if statements can be added around line pieces so that the template will only produce the necessary lines with no warnings.

RUNNING TEMPLATES:

These templates are all run using an SGRENDER statement. This is a simple statement that allows the user to identify the dataset to be used, the template to be run, and sets the values of any dynamic variables required. The dynamic variables can be set to text, numbers or variable names. Personally, I have run into issues sometimes using a variable name by itself, so I generally include with quotes – "'variable_name'n".

SGRENDER to run the horizontal bar chart:

```
proc sgrender data=metric_data1 template=b_style.baylor_hbar   ;
        dynamic         title1 = "Title Text"
                        title2  = "Title Text2"
                        foot1 = "Footnote Text 1"
                        foot2 =  "Footnote Text 2"
                        foot3 =  "Footnote Text 3"
                        foot4 = "Footnote Text 4"
                        _LOCATION_CODE='"location_code'n" /*variable name – y value*/
                        _METRIC_VALUE='"Metric_value'n"/*variable name – x value*/
                        _data_label ='"data_label'n" /*variable name – bar labels*/
                        _MAX = 100
                        _TARGETVAR1 = "'target'n";
        run;
```

CONCLUSION:

Creating complex graphics using GTL and the SG procedures is fairly simple and can be used to create simplified and easily supported graphics code. Using this process templates can be easily stored in a main location on a shared server and can be made available to all programmers. The templates shown here have been successfully used across multiple reports to control the look and simplify the process of report production.

REFERENCES:

Zender, Cynthia L. (2009). "Tiptoe through the templates." *Proceedings from SAS Global Forum 2009.* Available at https://support.sas.com/resources/papers/proceedings09/227-2009.pdf

Zender, Cynthia L. (2009). "SAS Style Templates: Always in Fashion." *Proceedings from SAS Global Forum 2010.* Available at http://support.sas.com/resources/papers/proceedings10/033-2010.pdf

Heath, Dan. (2009). "Secrets of the SG Procedures." *Proceedings from SAS Global Forum 2009.* Available at http://support.sas.com/resources/papers/proceedings09/324-2009.pdf

CONTACT INFORMATION:

Julie VanBuskirk
Nurtur Health, Inc.
jvanbuskirk@nurturhealth.com

ACKNOWLEDGEMENTS:

Special thanks go to the awesome group of programmers in the STEEEP Analytics department at Baylor Scott and White Health Care System. Thank you for the years of training and supporting the vision to create a reusable reporting infrastructure.

CODE:

Horizontal Bar:

```
proc template;
        define statgraph b_style.baylor_hbar_large;
        dynamic      TITLE1 TITLE2 FOOT1 FOOT2 FOOT3 FOOT4 _LOCATION_CODE2
                     _DISPLAY  _MAX   _LOCATION_CODE _METRIC_VALUE    _TARGETVAR1;
begingraph / backgroundcolor=CXFFFFFF   border=false;
        entrytitle TITLE1 / textattrs= (font= 'Arial' color= black size=12pt weight = bold)  ;
        entrytitle TITLE2 / textattrs= (font= 'Arial' color= black size=12pt) ;
        entryfootnote halign = right FOOT1  /textattrs=(font= 'Arial' color= dargr size=9pt);
        entryfootnote halign=left FOOT2 FOOT3 FOOT4 /textattrs=(font= 'Arial' color= dargr size=9pt );
 discreteattrvar attrvar=unitcolor var=_LOCATION_CODE2 attrmap="location_code" ;
  layout lattice;
    layout overlay /
            xaxisopts=(linearopts=(tickvalueformat=_DISPLAY    viewmin = 0 viewmax = _MAX)
                       display=(tickvalues line ticks)
                       tickvalueattrs = (font= 'Arial' color= dargr size=8pt)
                       offsetmin = 0  offsetmax = .80 )
            yaxisopts=( display=(line tickvalues )
                       tickvalueattrs =(font= 'Arial' color= dargr size=9pt))
            x2axisopts= (display= none  tickvalueattrs = (font= 'Arial' color= dargr size=9pt)
                       offsetmin= 0.2  offsetmax = 0  ) ;
          barchartparm x=_LOCATION_CODE y=_METRIC_VALUE  / yaxis=y
                    group=  unitcolor name='bar(h)' orient=horizontal;
           scatterplot x=xlabel y=_LOCATION_CODE  / markercharacter= _DATA_LABEL
                                                    markerattrs= (font= 'Arial' color= dargr size=9pt)
                                                    markercharacterattrs=(font= 'Arial' color= dargr
                                                   size=9pt)
                                                    xaxis=x2  ;
           scatterplot x=_TARGETVAR1 y=_LOCATION_CODE / markerattrs=(markersize = 9px
                                                          markersymbol = "diamondfilled")
                                                          markercolorgradient= traffic_flag
                                                          colormodel=threecolorramp;
   endlayout;   endlayout;
endgraph;
end;
run;
proc sgrender data=metric_data1 template=b_style.baylor_hbar   ;
        dynamic          title1 = "Title Text"
                         title2  = "Title Text2"
                         foot1 = "Footnote Text 1"
                         foot2 =  "Footnote Text 2"
                         foot3 =  "Footnote Text 3"
                         foot4 = "Footnote Text 4"
                         _LOCATION_CODE="'location_code'n" /*variable name – y value*/
                         _LOCATION_CODE2="'location_code'n" /*variable name – y value*/
                         _METRIC_VALUE="'Metric_value'n"/*variable name – x value*/
                         _data_label ="'data_label'n" /*variable name – bar labels*/
                         _MAX = 100
                          _TARGETVAR1 = "'target'n"
                         _DISPLAY = PERCENT8.2;
        run;
```

Control Chart:

```
proc template;
        define statgraph b_style.baylor_controlchart ;
        dynamic TITLE1 TITLE2 FOOT1 FOOT2 FOOT3 _YAXIS _DISPLAY  _DATE_VAR
                        _LABELVAR _MIN _MAX;
        begingraph /border=false ;
        entrytitle TITLE1 / textattrs= (font= 'Arial' color= black size=12pt weight = bold)  ;
        entrytitle TITLE2 / textattrs= (font= 'Arial' color= black size=12pt) ;
        entryfootnote halign = right FOOT1 /textattrs=(font= 'Arial' color= dargr size=9pt );
        entryfootnote halign=left FOOT2 FOOT3/textattrs=(font= 'Arial' color= dargr size=9pt);
        layout overlay /
                yaxisopts = (label = _YAXIS  /*y-axis label*/
                        display=(ticks tickvalues label line) /*display options*/
                        linearopts=(tickvalueformat=_DISPLAY viewmin = _MIN
                        viewmax =MAX)/*define  the tick value format */
                        tickvalueattrs = (font= 'Arial' color= dargr size=9pt ) /*define style options
                        for  tick values*/
                        labelattrs = Graphlabeltext )/*label style..this matches the color of the first
                        line*/
                /*define the graph area the x-axis takes up*/
                xaxisopts=  (display =( ticks tickvalues line)
                        tickvalueattrs = (font= 'Arial' color= dargr size=9pt )/*define style options for
                        tick values*/)/*x-axis display options*/;
        bandplot x=rep_period_date limitlower=_lower_ limitupper=_upper_ / display=(fill) type= step
                                        justify=center  extend=true fillattrs=(color=CXC2DEEA);
        referenceline y= ref1/ lineattrs= (color=gray);
        referenceline y= _targetvalue/ lineattrs= (color=green pattern= shortdash)
                                curvelabel= _targetlabel curvelabellocation=inside
                                curvelabelattrs=(color=green) curvelabelposition=max;
        seriesplot x= _date_var y= metric_value /display=all lineattrs= (color=black)
                        markerattrs= (markersize = 9px   markersymbol = "circlefilled");
        endlayout;
endgraph;
end;
run;
proc sgrender data=metric_data1 template=b_style.baylor_controlchart   ;
        dynamic         title1 = "Title Text"
                        title2  = "Title Text2"
                        foot1 = "Footnote Text 1"
                        foot2 =  "Footnote Text 2"
                        foot3 =  "Footnote Text 3"
                        _Yaxis = “Metric Label”
                        _DISPLAY = PERCENT8.2;
                        _Date_var = “’Report_period_date’n”
                        _LabelVar = “’Label_variable’n”
                        _MIN = 0
                        _MAX = 100
        run;
```