Paper 2988-2015

Building a Template from the Ground Up with GTL

Jedediah J. Teres, Verizon Wireless

ABSTRACT

This paper focuses on building a graph template in an easy-to-follow, step-by-step manner. The paper begins with using Graph Template Language to re-create a simple series plot, and then moves on to include a secondary y-axis as well as multiple overlaid block plots to tell a more complex and complete story than would be possible using only the SGPLOT procedure.

INTRODUCTION

SAS® Graph Template Language (GTL) is a very powerful and flexible tool for creating visualizations. However, making the move from exclusively using the SG procedures to designing one's own custom templates can be daunting. The flip side of the flexibility of GTL is that there are many options available. It can be overwhelming to the novice programmer. Even experienced SAS programmers who are new to GTL might benefit from having a starting point.

In this paper I lay out a step-by-step example based on a real world application to help demystify GTL. Specifically, the graph combines two different plot types, a SERIES plot and BLOCK plot to tell a more complete story than either plot alone.

Creating custom graph templates can be time-consuming, but the code can be reused and repurposed over and over again.

THE FINAL PRODUCT (SPOILER ALERT)

Rather than wait to reveal the graph I am building, I want to start with the end result. By presenting the code alongside the results, I hope to make clearer the steps involved and the purpose of the options used.

The graph in question shows several series plots against a block plot. It is two distinct plots presented together in one graph. The block plots classify different time periods, and have been colored in such a way as to allow for overlap and intersections of events. The series plots represent different metrics over time. Block plots can be created only with custom templates; there is no option in the SGPLOT procedure to create a block plot.

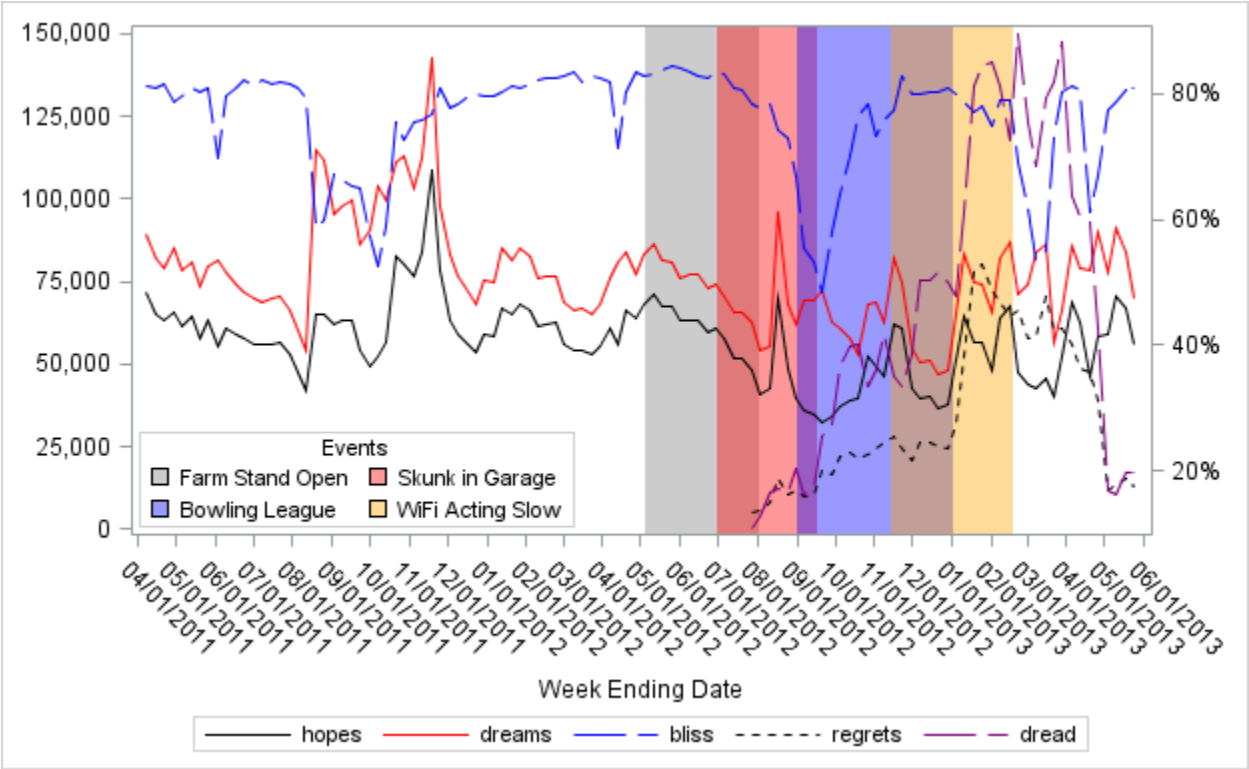
While this graph is based on a real-world example, the data presented herein have been altered to the point that they do not measure anything from the real world. The series plots could be stock prices, and the block plots could show the duration of different advertising campaigns. Or perhaps the series plots represent total sales and the block plots represent different product launch periods.

In order to make the concepts more universal, I am framing the series plots in existential terms, and the block plots correspond to events or activities a person might experience. It is my hope that this makes the material memorable without being a distraction to the reader.

The series plots are of "hopes," "dreams," "bliss," "regrets," and "dread." Bliss and Dread are expressed as proportions, while Hopes, Dreams, and Regrets are counts. In fact, Bliss is the ratio of Dreams to Hopes, while Dread is the ratio of Regrets to Hopes. I am not a philosopher and this is not meant to be taken too seriously.

Figure 1 shows the desired final product.

Figure 1



As mentioned above, this graph is two graphs in one. Figure 2 is a graph containing several series plots.

Figure 2

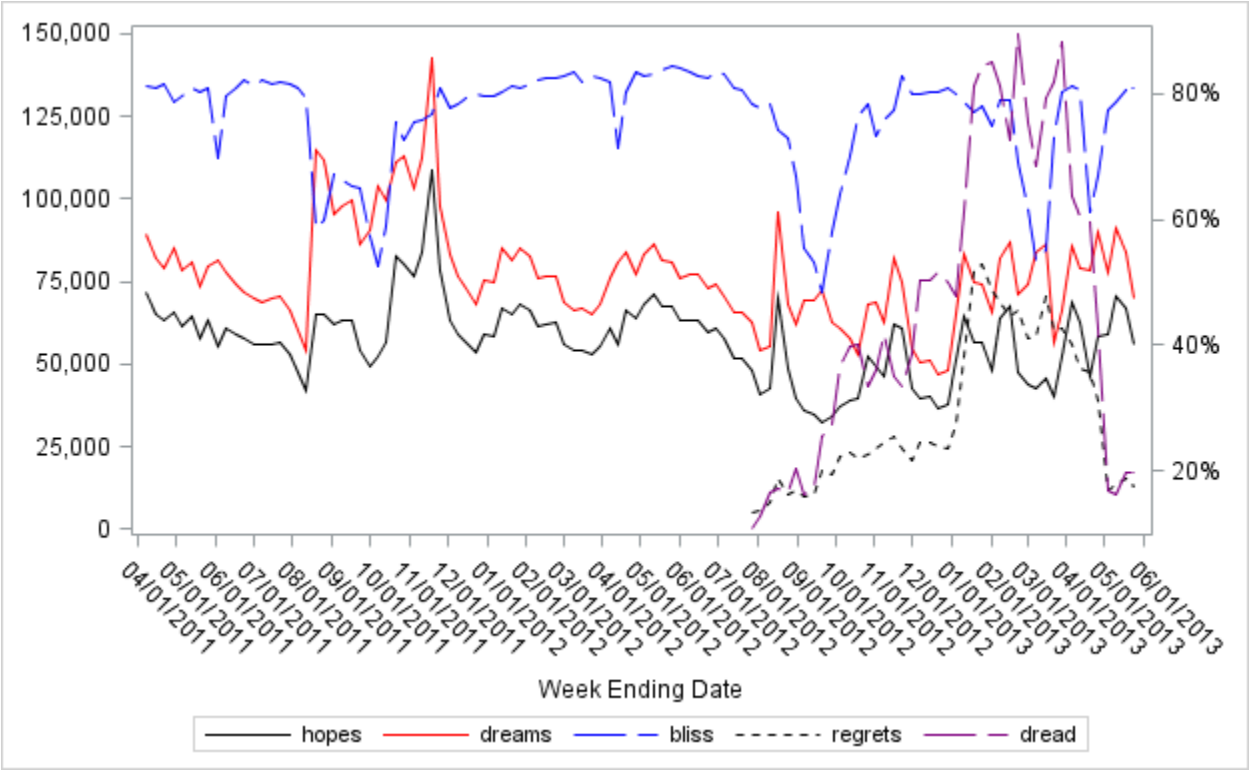
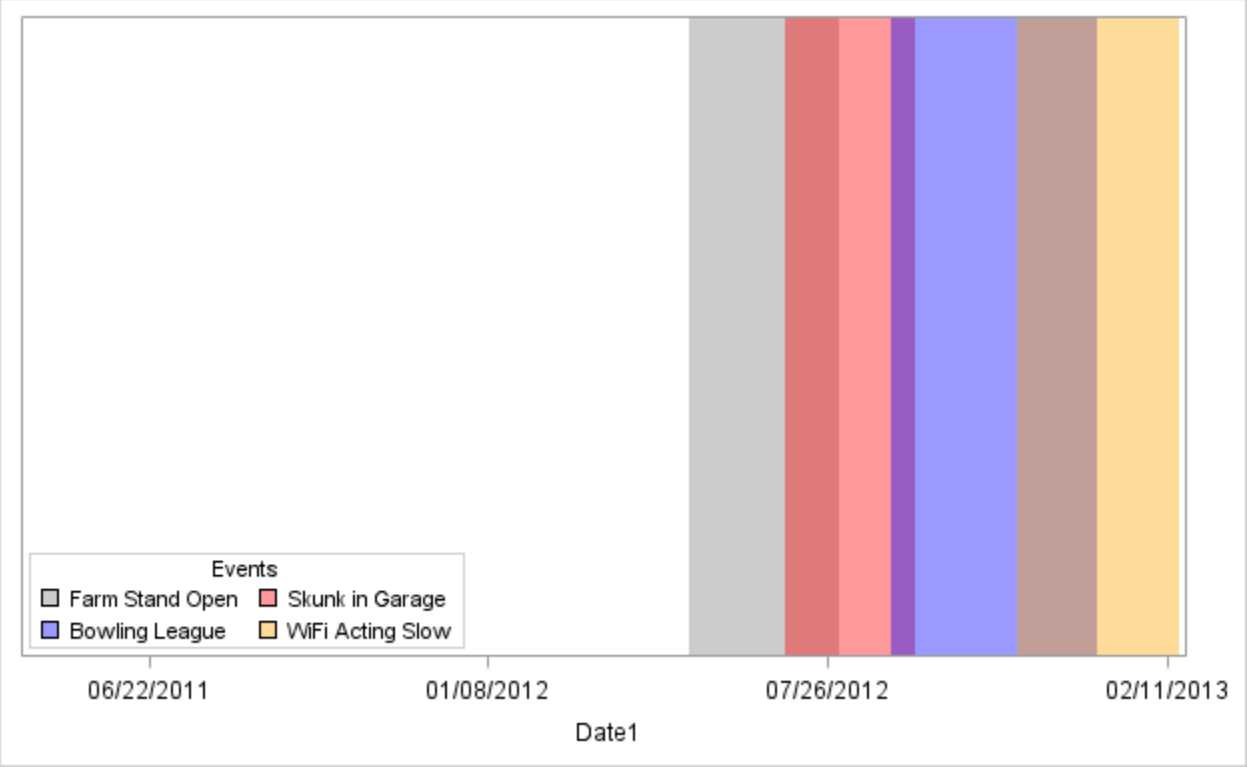


Figure 3 shows the block plots. The colors are somewhat transparent to allow for overlap. In this case, the grey bar represents the time during which a particular farm stand was open. The light red bar represents the period of time during which a skunk was living in the garage, and the darker red bar corresponds to the period of time during which the farm stand was open *and* the skunk was living in the garage. Similarly, the lavender strip shows the part of bowling league season when the Wi-Fi was acting slow.

Figure 3



THE SERIES PLOT

The first step in building up the template is creating the series plots.

First, I present a brief primer on the difference between using GTL to create graphics and using other graphing procedures such as SGPLOT.

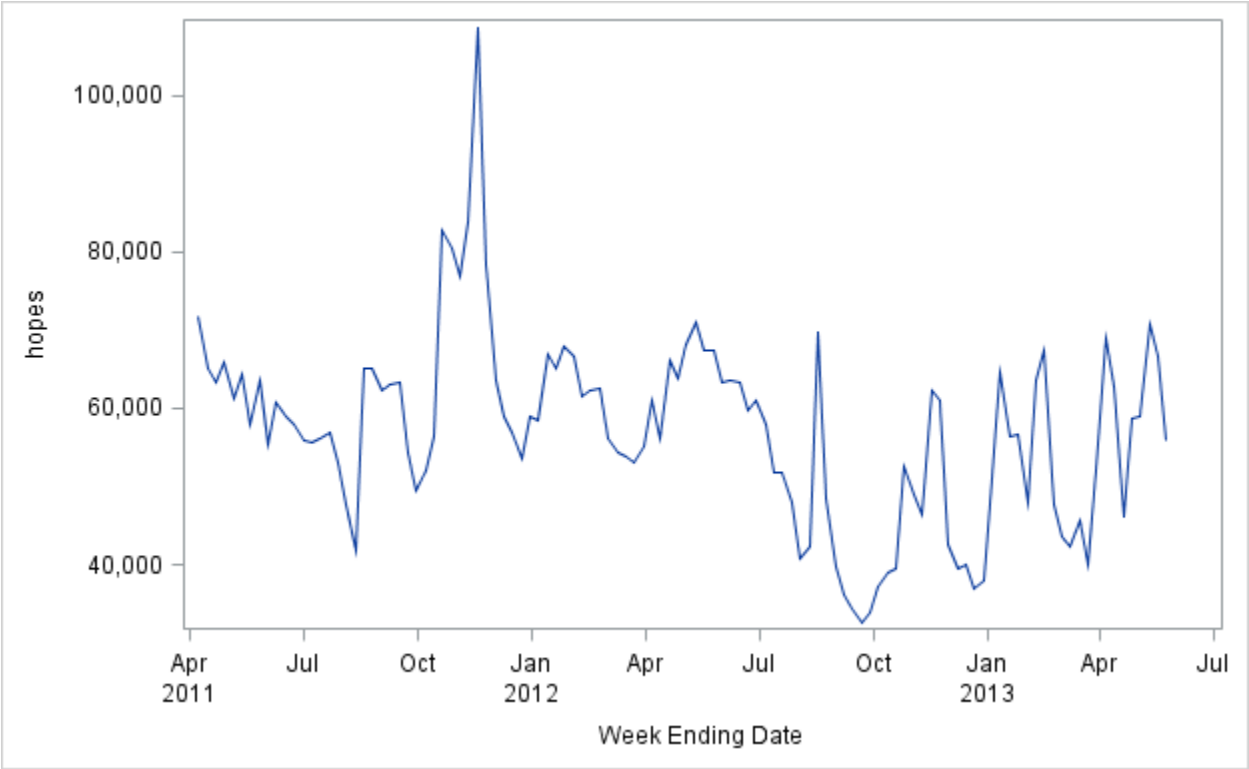
Running PROC SGPLOT creates a graph. Graph templates are written using the TEMPLATE procedure. However, running PROC TEMPLATE does not produce a graph. Rather, it produces a template that is used with the SGRENDER procedure. It is PROC SGRENDER that produces the actual graph.

Before delving into the specifics of PROC TEMPLATE and GTL, consider the code to create a series plot using PROC SGPLOT:

```
proc sgplot data = SGF_graph_final ;
     series    x = week_ending
               y = hopes ;
run ;
```

The resulting graph follows in Figure 4.

Figure 4



Compare the PROC SGPLOT code to the GTL code:

```
proc template ;
     Define StatGraph Series0 ;
          BeginGraph ;
               Layout Overlay ;
                    SeriesPlot x = week_ending
                               y = hopes ;
               EndLayout ;
          EndGraph ;
     End ;
run ;
```

The innermost section of code is nearly identical to the PROC SGPLOT code used earlier. The only difference is that the SERIES statement in PROC SGPLOT becomes SERIESPLOT using GTL.

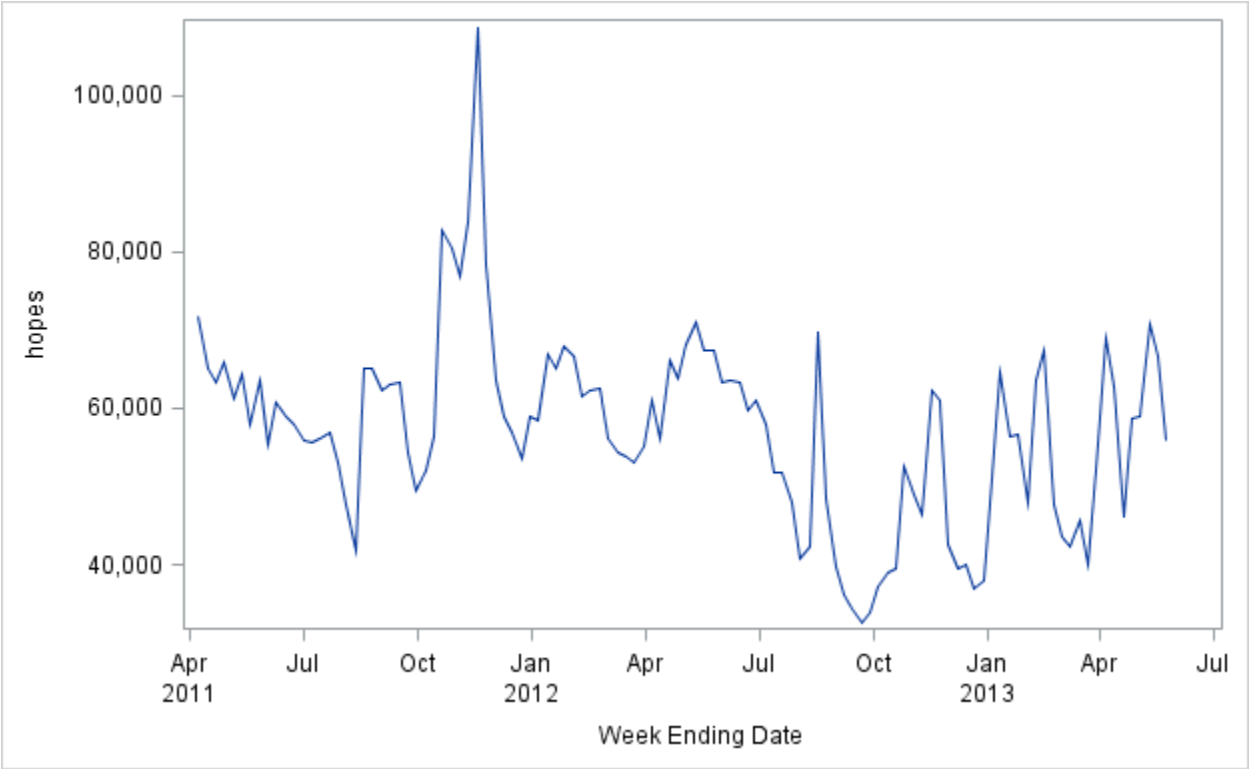
The PROC TEMPLATE code consists of three levels. The outermost level is the DEFINE statement, which takes a closing END statement. We are defining a STATGRAPH. Next is the BEGINGRAPH statement, paired with an ENDGRAPH closing statement. The GTL code goes in this section. Then we have the LAYOUT statement. The code between the LAYOUT statement and the ENDLAYOUT statement determines what will go into the graph and how it will appear.

Recall that PROC TEMPLATE does not produce a graph. It produces a template. In this case, the template is called "Series0." In order to create a graph using this template, we use PROC SGRENDER with the Series0 template as shown below:

```
proc sgrender data     = mydocs.SGF_graph_final
              template = Series0 ;
run ;
```

Figure 5 (made with PROC SGRENDER) is identical to Figure 4 (made with PROC SGPLOT).

Figure 5



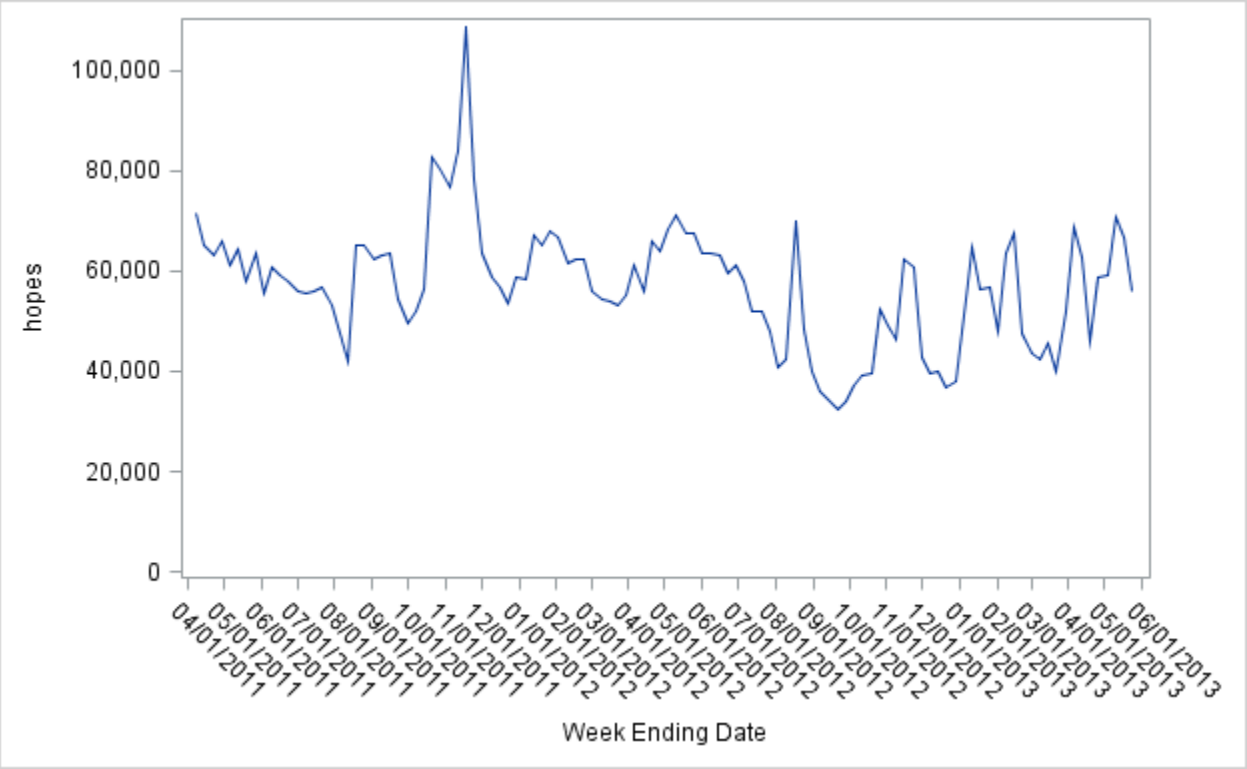
Having established a starting point, creating the template to create the final series plot is merely a matter of refining and customizing our template code.

The first refinement is adding options to specify how the axes will appear. Our x-axis is time based, and so we specify that the INTERVAL should be a month and displayed in the MMDDYY10 format. The TICKVALUEFITPOLICY determines how SAS will orient the tick labels. In this case, they are rotated and have minimal spacing (ROTATETHIN). We also specify a minimum value of 0 for the y-axis with the VIEWMIN option. Note that the axis options (XAXISOPTS and YAXISOPTS, respectively) are part of the LAYOUT statement and are preceded by a slash (\):

```
proc template ;
  Define StatGraph Series1 ;
    BeginGraph ;
      Layout Overlay /
          xaxisopts = (                                  /* new */
               TimeOpts  = (                             /* new */
                    interval = month                     /* new */
                    TickValueFormat = mmddyy10.          /* new */
                    TickValueFitPolicy  = RotateThin)) /* new */
          yaxisopts = (                                  /* new */
               LinearOpts = (ViewMin = 0)) ;             /* new */

          SeriesPlot x = week_ending
                     y = hopes ;
      EndLayout ;
    EndGraph ;
  End ;
run ;
```

The SGRENDER code has been omitted as it does not change. The resulting graph is shown in Figure 6. Note the x-axis format has changed, and the y-axis starts at 0 as per our specification.

Figure 6



At this point, adding additional series plots is very straightforward—it's simply a matter of including addition SERIESPLOT statements.

However, with multiple series sharing the same graph, it's important to specify distinct line attributes to clearly differentiate the different series. Using the LINEATTRS option, the color and pattern to be used for each series can specified. In general, it's a good idea to explicitly specify the key attributes of your graphics because the different output destinations (RTF, HTML, PDF) have different defaults and the same graph may appear differently in different output destinations otherwise.

The y-axis has been labeled using the name of the Y variable. This is default behavior that must be changed, since this label would be misleading with multiple series plotted. This is done using a DISPLAY option in the YAXISOPTS statement to explicitly list the attributes to be shown on the axis.

Since the y-axis will no longer be labeled, a legend should be added to clearly label each series. To include a legend, the DISCRETELEGEND statement is used. A NAME must be assigned to each series in order to include it in the legend. The names assigned to the series plots are then included in the DISCRETELEGEND statement as show below.

The placement of the legend is specified by the LOCATION, VALIGN, and HALIGN options, all of which are part of the DISCRETELEGEND statement. In this case, the LOCATION is OUTSIDE, meaning it will appear outside of the main graph area. The VALIGN and HALIGN values are set to BOTTOM and CENTER, respectively, which (predictably) places the legend on the bottom and in the center.

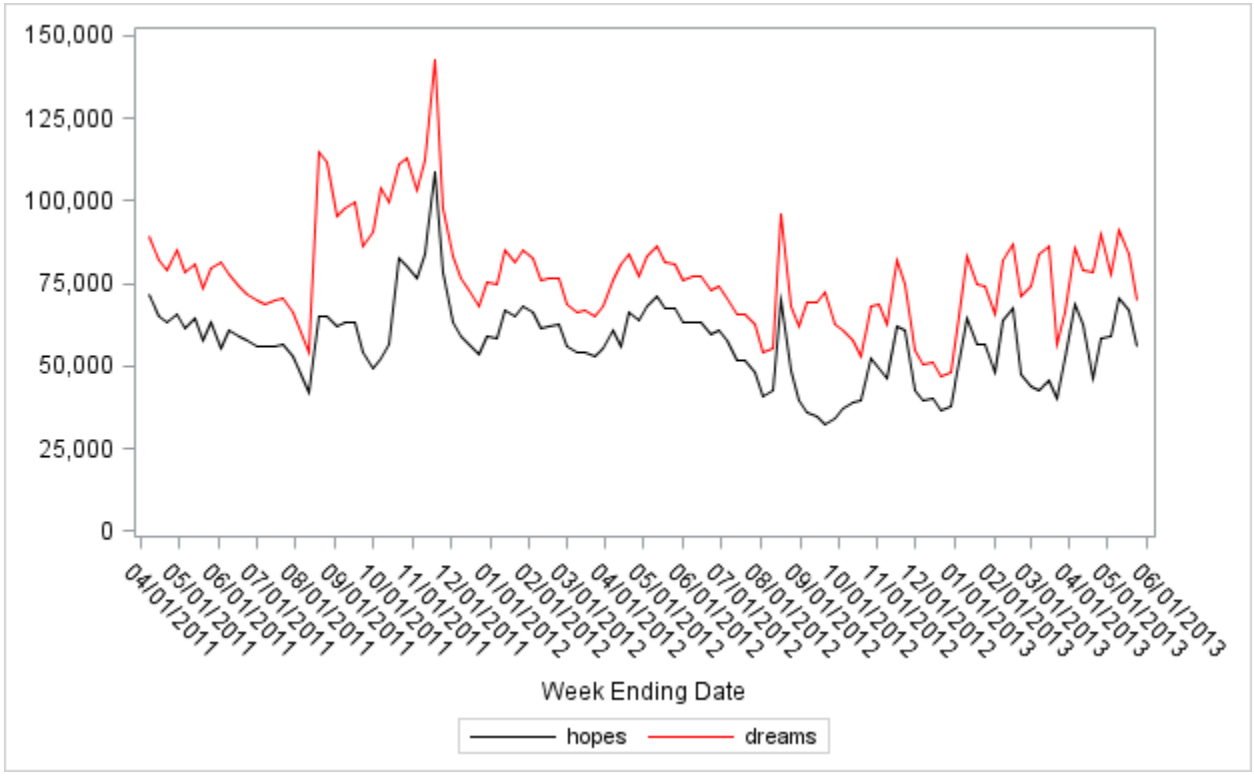
Additions and changes to the previous PROC TEMPLATE code are marked with comments. The statements preceding and following the LAYOUT block have been excluded because they have not changed. The XAXISOPTS have been omitted as well.

The revised template code follows on the next page:

```
proc template ;
   [DEFINE and BEGINGRAPH statements omitted]
      Layout Overlay /
        xaxisopts = [option statements omitted]
        yaxisopts = (LinearOpts = (ViewMin = 0)
                     Display    = (Ticks TickValues)) ;      /* new */

        SeriesPlot x = week_ending
                   y = hopes /
          name      = 's1'                    /* new */
          lineattrs = (color   = black        /* new */
                       pattern = solid) ;     /* new */

        SeriesPlot x = week_ending
                   y = dreams /
          name      = 's2'                    /* new */
          lineattrs = (color   = red          /* new */
                       pattern = solid) ;     /* new */

       DiscreteLegend 's1' 's2' / Location  = Outside  /* new */
                                  Valign    = Bottom   /* new */
                                  Halign    = Center ; /* new */
      EndLayout ;
  [ENDGRAPH and END statements omitted]
run ;
```

Figure 7 shows the graph with the added series formats and legend produced using the revised template.

Figure 7

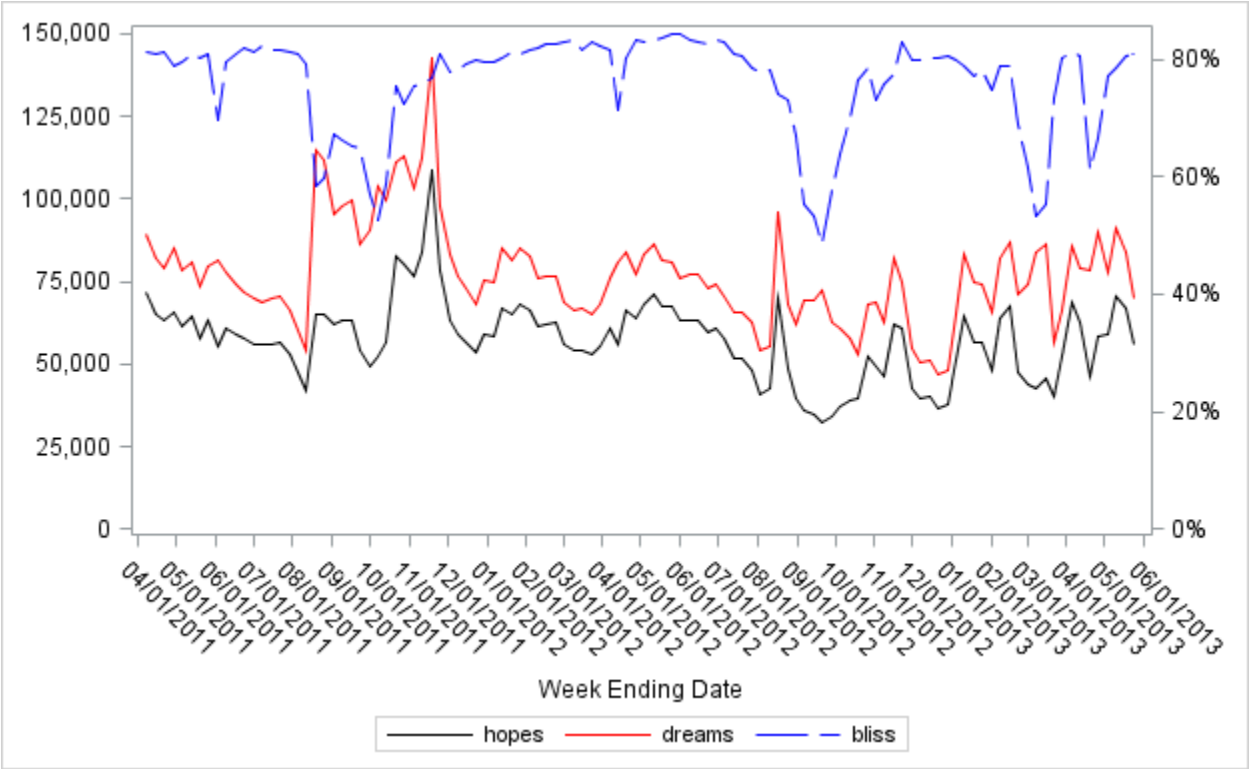


If additional series will be plotted using a different scale, then a secondary y-axis is required. In this example, Bliss and Dread are proportions and so an appropriately scaled y-axis is required for these metrics. Note that GTL code that has not changed is omitted.

```
proc template ;
   [DEFINE and BEGINGRAPH statements omitted]
      Layout Overlay /
        xaxisopts = [options omitted] ;
        yaxisopts = [options omitted] ;

        y2axisopts = (                              /* new */
          LinearOpts = (ViewMin = 0)                /* new */
          Display    = (Ticks TickValues)) ;        /* new */

        SeriesPlot x = week_ending
                   y = hopes /   [options omitted] ;

        SeriesPlot x = week_ending
                   y = dreams /  [options omitted] ;

        SeriesPlot x = week_ending                  /* new */
                   y = bliss /                      /* new */
          name      = 's3'                          /* new */
          yaxis     = y2                            /* new- note Y2 */
          lineattrs = (color   = blue               /* new */
                       pattern = LongDash) ;        /* new */

       [DISCRETELEGEND statement omitted]
      EndLayout ;
   [ENDGRAPH and END statements omitted]
run ;
```

Figure 8 is the graph produced by the revised template that includes the secondary y-axis plot.

Figure 8



Now that the secondary y-axis has been established, creating the final series plot template is simply a matter of adding two more SERIESPLOT statements. The PROC TEMPLATE code to generate the final series plot is in Appendix A.

THE BLOCK PLOT

All that remains is building the block plot template. Some of the techniques described earlier are applicable, but the block plot is a very different beast.

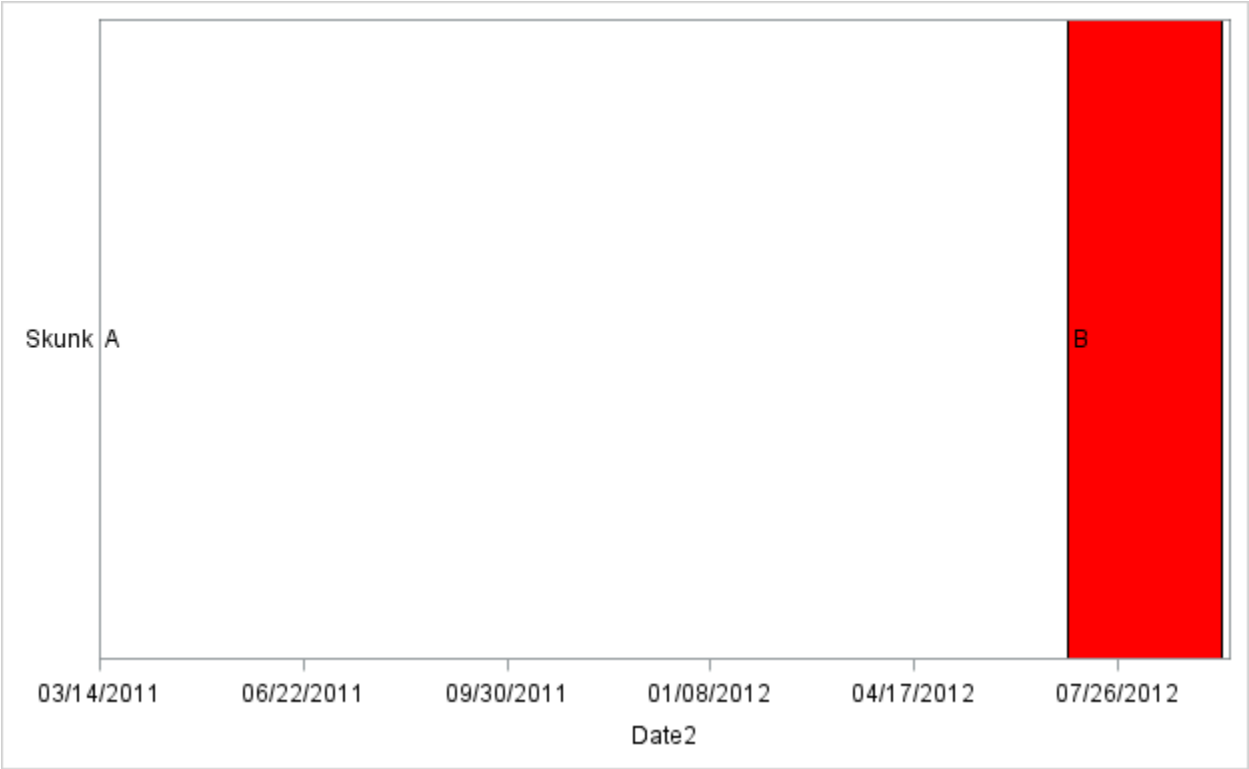
Block plots, like other plots, expect the data to be in a specific format. Here is a data step creating the underlying data for a single block plot:

```
data Skunk ;
     input Date2 date9. Skunk $2. ;
     format Date2 mmddyy10. ;
     datalines ;
08APR2011 A
01JUL2012 B
15SEP2012 C
;
run ;
```

The data set Skunk has two variables, date2 and skunk. There are three observations.

Figure 9 shows a block plot created with the Skunk data set created above. Note the relationship between the values of date2 and skunk with the different sections of the plot.

Figure 9



The template used to create this plot follows:

```
proc template ;
  define StatGraph Block1 ;
    BeginGraph ;
      Layout Overlay ;
        BlockPlot x       = date2
                  block   = Skunk /
          Display         = (fill label outline values)
          FillType        = alternate
          FillAttrs       = (color = white)
          AltFillAttrs    = (color = red) ;
      EndLayout ;
    EndGraph ;
  End ;
run ;
```

Note that the PROC TEMPLATE code is very similar to the code used to create the series plot templates. The only difference is that a BLOCKPLOT statement is used rather than a SERIESPLOT.

The FILLTYPE option of the block plot is set to ALTERNATE. The COLOR of the FILLATTRS is set to WHITE and the COLOR of the ALTFILLATTRS is set to RED, so what appears to be a lone red stripe in Figure 9 is actually an alternating series of red and white stripes. This is an important distinction.

In this example, each block is defined by three data points- the starting point of the first section, the starting point of the second section (and implicitly, the ending point of the first section), and the starting point of the third section (and ending point of the third section). Adding more data points would create additional blocks with the specified alternating pattern.

Including additional block plots is not as simple as including multiple series plots. Based on the approach taken with the series plots above, one might write code similar to this:

```
proc template ;
   [DEFINE and BEGINGRAPH statements omitted]
      Layout Overlay ;
        BlockPlot x     = date2
                  block = Skunk /
          Display      = (fill)
          FillType     = alternate
          FillAttrs    = (color = white)
          AltFillAttrs = (color = red) ;

        BlockPlot x     = date3
                  block = Bowling /
          Display      = (fill)
          FillType     = alternate
          FillAttrs    = (color = white)
          AltFillAttrs = (color = blue) ;
      EndLayout ;
   [ENDGRAPH and END statements omitted]
run ;
```

Figure 10

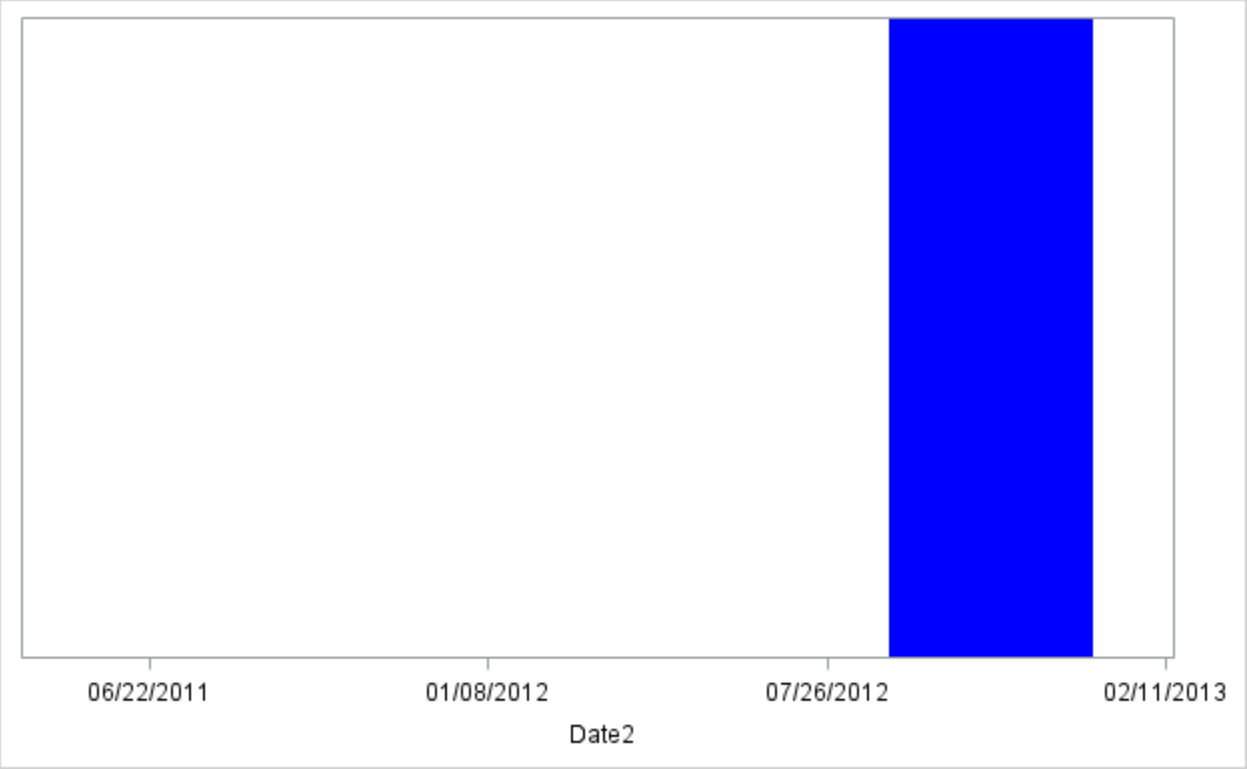
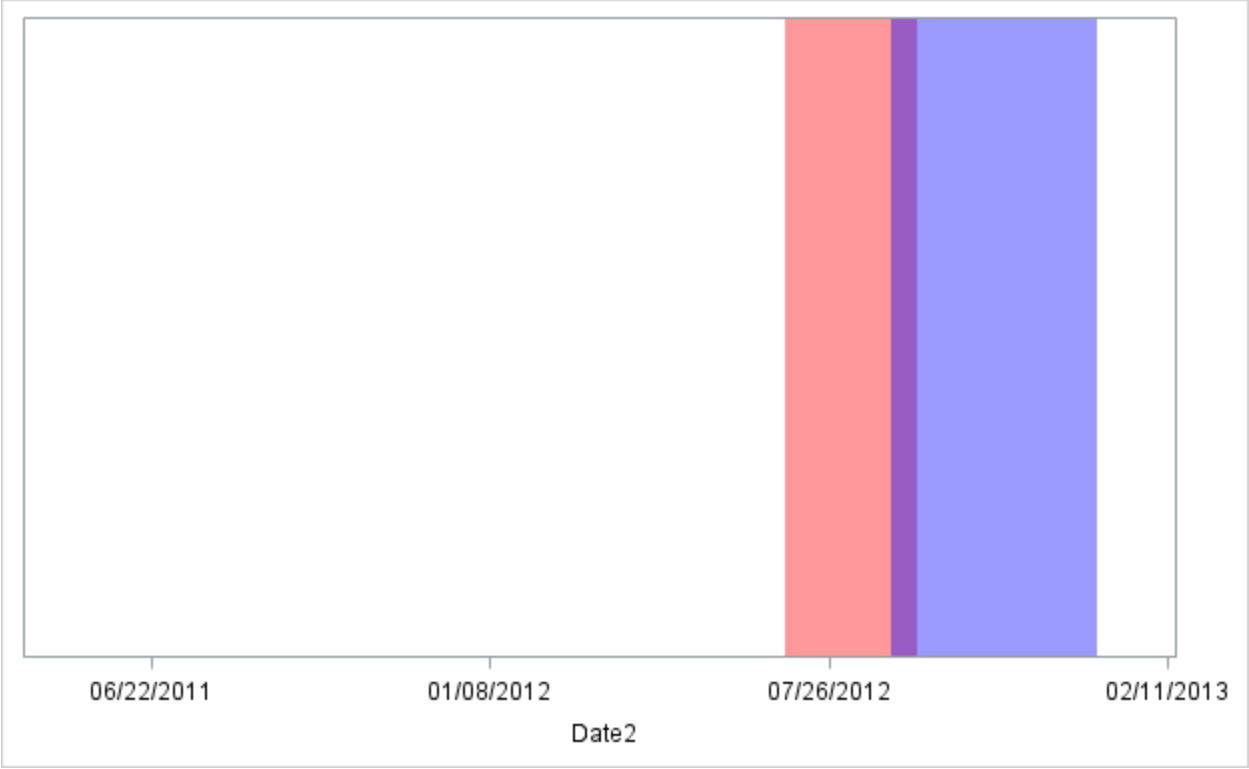


Figure 10 shows only one block. The key to plotting multiple blocks lies with the fact that each of the fill attributes associated with an alternating block plot (FILLATTRS and ALTFILLATTRS) has independent transparency settings. The TRANSPARENCY for the white sections can be set to 1 (i.e., completely transparent) and the TRANSPARENCY for the red and blue sections can be set to .6, as seen below:

```
proc template ;
   [DEFINE and BEGINGRAPH statements omitted]
      Layout Overlay ;
        BlockPlot x     = date2
                  block = Skunk /
          Display      = (fill)
          FillType     = alternate
          FillAttrs    = (color        = white
                          transparency = 1)      /* new */
          AltFillAttrs = (color        = red
                          transparency = .6) ;   /* new */

        BlockPlot x     = date3
                  block = Bowling /
          Display      = (fill)
          FillType     = alternate
          FillAttrs    = (color        = white
                          transparency = 1)      /* new */
          AltFillAttrs = (color        = blue
                          transparency = .6) ;   /* new */
      EndLayout ;
   [ENDGRAPH and END statements omitted]
run ;
```

Figure 11 shows the overlapping block plots. The red section is when the skunk was in the garage, while the blue section shows the bowling league period. The purple stripe is when both events overlapped.

Figure 11



Adding a legend for a block plot is a little trickier than it was with the series plot legend. When the ALTERNATE option is selected for FILLTYPE, then the block plot does not support a DISCRETELEGEND entry. So we have to build a legend by defining a series of individual LEGENDITEMs that we will then supply to a DISCRETELEGEND statement.

To do so, a LEGENDITEM is created corresponding to each block. It is assigned a name and given a LABEL. In order to use TRANSPARENCY, the TYPE is set to FILL[1]. Note that the LEGENDITEM statements are placed *before* the LAYOUT statement.

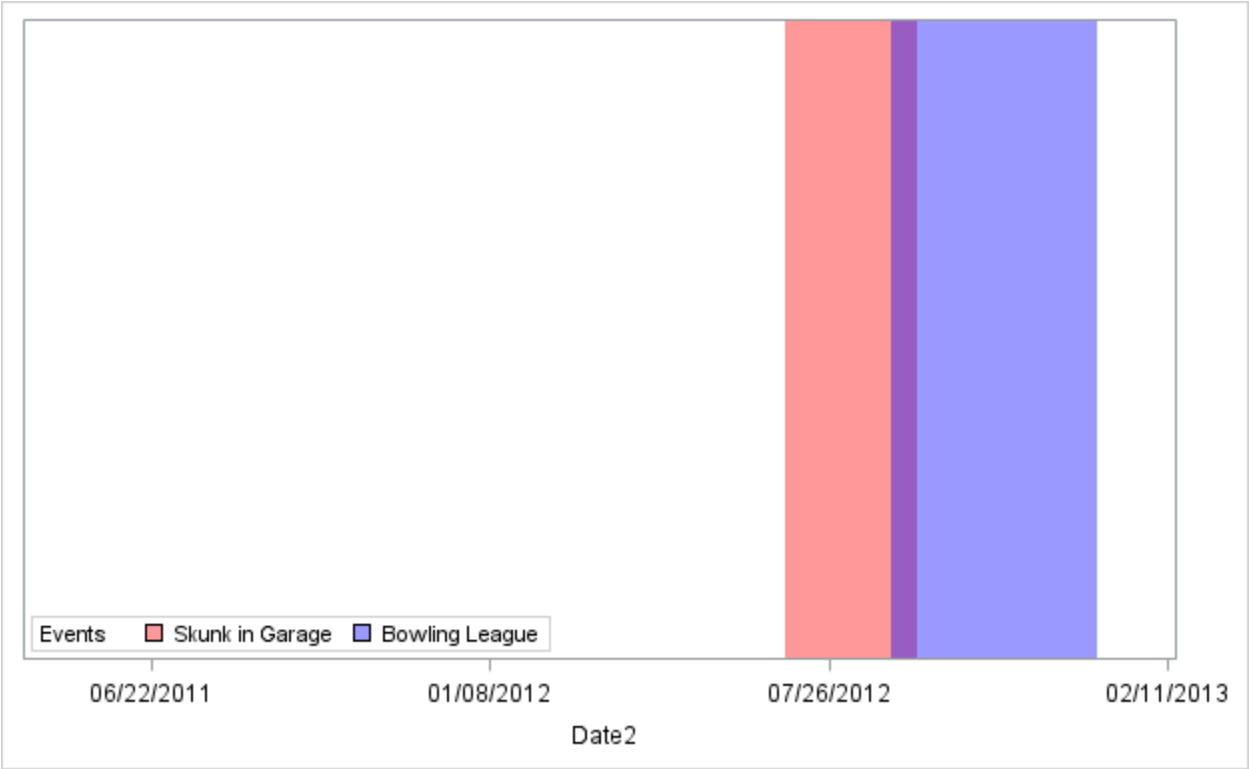
The DISCRETELEGEND statement references the LEGENDITEMs. The LOCATION is set to INSIDE so that the legend will appear inside the plot area. The VALIGN and HALIGN are set to BOTTOM and LEFT, respectively, so the legend will appear in the bottom left of the plot area. One new option is used—ACROSS is set to "2," meaning the legend will have no more than two columns, regardless of how many entries. A TITLE has also been specified.

```
proc template ;
  [DEFINE and BEGINGRAPH statements omitted]
      LegendItem  name = 'b2'           /* new */
                  type = Fill /         /* new */
        Label     = 'Skunk in Garage' /* new */
        FillAttrs = (                   /* new */
          Color        = Red            /* new */
          Transparency = .6) ;          /* new */

      LegendItem name = 'b3'            /* new */
                 type = Fill /          /* new */
        Label     = 'Bowling League'    /* new */
        FillAttrs = (                   /* new */
          Color        = Blue           /* new */
          Transparency = .6) ;          /* new */

      Layout Overlay ;
        BlockPlot x     = date2
                  block = Skunk /     [OPTIONS OMITTED] ;

        BlockPlot x     = date3
                  block = Bowling /   [OPTIONS OMITTED] ;

        ** Block Plot Legend ;
        DiscreteLegend 'b2' 'b3' /
          Title    = "Events"
          Across   = 2
          Location = Inside
          VAlign   = Bottom
          HAlign   = Left ;
      EndLayout ;
    EndGraph ;
  End ;
run ;
```

Figure 12 shows the graph with its new legend.

[1] In SAS 9.4, transparency has been added as an option for the MARKER TYPE, but this code precedes that release.

Figure 12

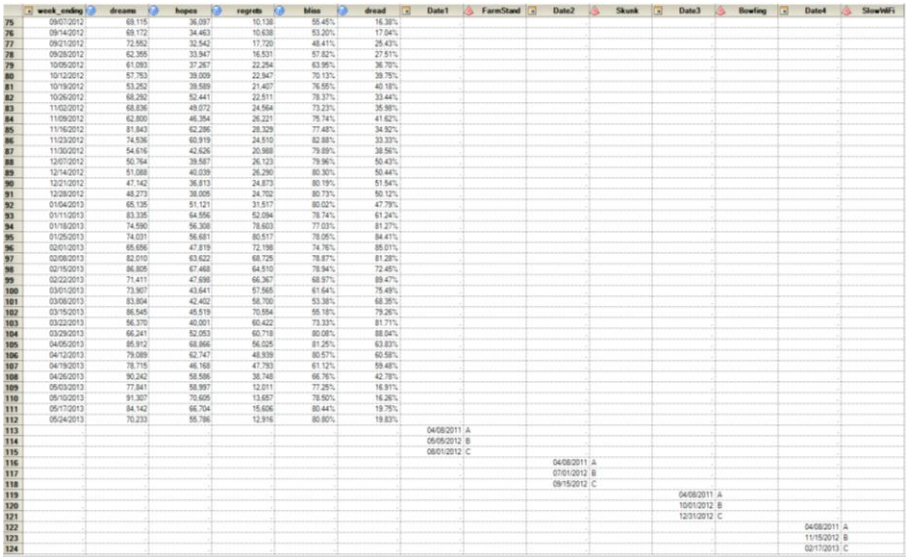


Adding more blocks to the graph is straightforward. The final block plot template code appears in Appendix B.

REGARDING BLOCK PLOT DATA

Each block plot takes its own data set. To use a single data set to create many block plots, the individual block plot data sets need to be concatenated rather than merged. The resulting shape of the data used for graphing may not be intuitive, so it bears mentioning. Merging the concatnated block plot data with the series plot creates the data set depicted in Figure 13.**Error! Reference source not found.** The series plot data is in the upper left, and the block plot data is in the lower right.

Figure 13



THE FINAL PLOT (REVISITED)

Now that the individual templates for the series plots and the block plots are complete, combining them into one template is easy. The code appears in Appendix C.

CONCLUSION

Custom Graph Templates can be used to make complex graphs that would not be possible to create using only the standard SG procedures. Combines different plot types can tell a richer and more complete story. In the example presented here, the block plots provide context for some of the fluctuations observed in the series plots.

When combining different plot types, it's important to understand the structure of the underlying data. The input data set needs to support all the plots used in the graph. Not all plot types can be combined. A list of incompatible types is beyond the scope of this paper.

Fine-tuning graphing templates can take some degree of trial and error. However, once a template has been created and finalized, it can be used again and again.

The Graph Template Language is very powerful and flexible, and includes the ability to parameterize the template code using the DYNAMIC statement. An example of using dynamics is beyond the scope of this paper, but it's the next step towards mastering GTL.

REFERENCES

SAS Institute 2012. *SAS® 9.3 Graph Template Language: Reference, Third Edition*. Cary, NC: SAS Institute Inc.

SAS Institute 2012. *SAS® 9.3 Output Delivery System: User's Guide, Second Edition.* Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

The author wishes to thank Sanjay Matange for his assistance in solving the block plot transparency problem via the SAS Community discussion forum.

RECOMMENDED READING

- *Secrets of the SG Procedures*
- *Introduction to the Graph Template Language*
- *Off the Beaten Path: Create Unusual Graphs with GTL*
- *Using SAS® GTL to Visualize Your Data When There is Too Much of It to Visualize*
- *A Programmer's Introduction to the Graphics Template Language*
- *Free Expressions and Other GTL Tips*

CONTACT INFORMATION

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

Jedediah J. Teres
Verizon Wireless
jedediah.teres@verizonwireless.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

APPENDIX A: SERIES PLOT GRAPH TEMPLATE

```
proc template ;
  Define StatGraph FinalSeries ;
     BeginGraph ;

        Layout Overlay /
           xaxisopts  = (TimeOpts  = (
              interval            = month
              TickValueFormat     = mmddyy10.
              TickValueFitPolicy  = RotateThin))

           yaxisopts  = (Display   = (Ticks TickValues))

           y2axisopts = (Display   = (Ticks TickValues)) ;

           SeriesPlot x = week_ending
                      y = Hopes    / name      = 's1'
                                     lineattrs = (color   = black
                                     pattern   = solid) ;

           SeriesPlot x = week_ending
                      y = Dreams   / name      = 's2'
                                     lineattrs = (color   = red
                                     pattern   = solid) ;

           SeriesPlot x = week_ending
                      y = Bliss    / name      = 's3'
                                     yaxis     = y2
                                     lineattrs = (color   = blue
                                     pattern   = LongDash) ;

           SeriesPlot x = week_ending
                      y = Regrets / name      = 's4'
                                     lineattrs = (color   = black
                                     pattern   = ShortDash) ;

           SeriesPlot x = week_ending
                      y = Dread    / name = 's5'
                                     yaxis     = y2
                                     lineattrs = (color   = purple
                                                  pattern = LongDash) ;

           DiscreteLegend 's1' 's2' 's3' 's4' 's5' /
              Location  = Outside
              Valign    = Bottom
              Halign    = Center ;

        EndLayout ;
     EndGraph ;
  End ;
run ;
```

APPENDIX B: BLOCK PLOT GRAPH TEMPLATE

```
proc template ;
  Define StatGraph FinalBlock ;
    BeginGraph ;

      LegendItem name = 'b1'
                 type = Fill / Label     = 'Farm Stand Open'
                               FillAttrs = (Color        = Gray
                                            Transparency = .6) ;

      LegendItem name = 'b2'
                 type = Fill / Label     = 'Skunk in Garage'
                               FillAttrs = (Color        = Red
                                            Transparency = .6) ;

      LegendItem name = 'b3'
                 type = Fill / Label     = 'Bowling League'
                               FillAttrs = (Color        = Blue
                                            Transparency = .6) ;

      LegendItem name = 'b4'
                 type = Fill / Label     = 'WiFi Acting Slow'
                               FillAttrs = (Color        = Orange
                                            Transparency = .6) ;

      Layout Overlay ;

        BlockPlot x     = date1
                  block = FarmStand / Display      = (fill)
                                      FillType     = alternate
                                      FillAttrs    = (
                                        color        = white
                                        transparency = 1)
                                      AltFillAttrs = (
                                        color        = gray
                                        transparency = .6) ;

        BlockPlot x     = date2
                  block = Skunk     / Display      = (fill)
                                      FillType     = alternate
                                      FillAttrs    = (
                                        color        = white
                                        transparency = 1)
                                      AltFillAttrs = (
                                        color        = red
                                        transparency = .6) ;
```

[continued on following page]

[continued from previous page]

```
        BlockPlot x     = date3
                  block = Bowling   / Display      = (fill)
                                      FillType     =  alternate
                                      FillAttrs    = (
                                        color        = white
                                        transparency = 1)
                                      AltFillAttrs = (
                                        color        = blue
                                        transparency = .6) ;

        BlockPlot x     = date4
                  block = SlowWiFi  / Display      = (fill)
                                      FillType     = alternate
                                      FillAttrs    = (
                                        color        = white
                                        transparency = 1)
                                      AltFillAttrs = (
                                        color        = orange
                                        transparency = .6) ;

        DiscreteLegend 'b1' 'b2' 'b3' 'b4' /
          Title    = "Events"
          Across   = 2
          Down     = 2
          Location = Inside
          VAlign   = Bottom
          HAlign   = Left ;

      EndLayout ;

    EndGraph ;
  End ;
run ;
```

APPENDIX C COMBINED SERIES AND BLOCK PLOT GRAPH TEMPLATE

```
proc template ;
  Define StatGraph FinalSeriesBlock ;
     BeginGraph ;

      LegendItem name = 'b1'
                 type = Fill / Label     = 'Farm Stand Open'
                               FillAttrs = (Color        = Gray
                                            Transparency = .6) ;

      LegendItem name = 'b2'
                 type = Fill / Label     = 'Skunk in Garage'
                               FillAttrs = (Color        = Red
                                            Transparency = .6) ;

      LegendItem name = 'b3'
                 type = Fill / Label     = 'Bowling League'
                               FillAttrs = (Color        = Blue
                                            Transparency = .6) ;

      LegendItem name = 'b4'
                 type = Fill / Label     = 'WiFi Acting Slow'
                               FillAttrs = (Color        = Orange
                                            Transparency = .6) ;

        Layout Overlay /
           xaxisopts  = (TimeOpts  = (
              interval            = month
              TickValueFormat     = mmddyy10.
              TickValueFitPolicy  = RotateThin))

           yaxisopts  = (Display   = (Ticks TickValues))

           y2axisopts = (Display   = (Ticks TickValues)) ;

           SeriesPlot x = week_ending
                      y = Hopes    / name      = 's1'
                                     lineattrs = (color    = black
                                     pattern   = solid) ;

           SeriesPlot x = week_ending
                      y = Dreams   / name      = 's2'
                                     lineattrs = (color    = red
                                     pattern   = solid) ;

           SeriesPlot x = week_ending
                      y = Bliss    / name      = 's3'
                                     yaxis     = y2
                                     lineattrs = (color    = blue
                                     pattern   = LongDash) ;
```

[continued on following page]

[continued from previous page]

```
            SeriesPlot x = week_ending
                       y = Regrets / name      = 's4'
                                     lineattrs = (color   = black
                                     pattern   = ShortDash) ;

            SeriesPlot x = week_ending
                       y = Dread   / name = 's5'
                                     yaxis     = y2
                                     lineattrs = (color   = purple
                                                  pattern = LongDash) ;

            BlockPlot x     = date1
                      block = FarmStand / Display      = (fill)
                                          FillType     = alternate
                                          FillAttrs    = (
                                            color        = white
                                            transparency = 1)
                                          AltFillAttrs = (
                                            color        = gray
                                            transparency = .6) ;

            BlockPlot x     = date2
                      block = Skunk     / Display      = (fill)
                                          FillType     = alternate
                                          FillAttrs    = (
                                            color        = white
                                            transparency = 1)
                                          AltFillAttrs = (
                                            color        = red
                                            transparency = .6) ;
            BlockPlot x     = date3
                      block = Bowling   / Display      = (fill)
                                          FillType     =  alternate
                                          FillAttrs    = (
                                            color        = white
                                            transparency = 1)
                                          AltFillAttrs = (
                                            color        = blue
                                            transparency = .6) ;

            BlockPlot x     = date4
                      block = SlowWiFi  / Display      = (fill)
                                          FillType     = alternate
                                          FillAttrs    = (
                                            color        = white
                                            transparency = 1)
                                          AltFillAttrs = (
                                            color        = orange
                                            transparency = .6) ;
```

[continued on following page]

[continued from previous page]

```
            DiscreteLegend 's1' 's2' 's3' 's4' 's5' /
               Location  = Outside
               Valign    = Bottom
               Halign    = Center ;

            DiscreteLegend 'b1' 'b2' 'b3' 'b4' /
              Title    = "Events"
              Across   = 2
              Down     = 2
              Location = Inside
              VAlign   = Bottom
              HAlign   = Left ;

         EndLayout ;

      EndGraph ;
   End ;
run ;
```