Great Time to Learn GTL:
A Step-by-Step Approach to Creating the Impossible

Richann Watson, DataRich Consulting

ABSTRACT

Output Delivery System (ODS) graphics, produced by SAS® procedures, are the backbone of the Graph Template Language (GTL). Procedures such as the Statistical Graphics (SG) procedures dynamically generate GTL templates based on the plot requests made through the procedure syntax. For this paper, these templates will be referenced as procedure-driven templates. GTL generates graphs using a template definition that provides extensive control over output formats and appearance. Would you like to learn how to build your own template and make customized graphs and how to create that one highly desired, unique graph that at first glance seems impossible? Then it's a **G**reat **T**ime to **L**earn GTL! This paper guides you through the GTL fundamentals while walking you through creating a graph that at first glance appears too complex but is truly simple once you understand how to build your own template.

INTRODUCTION

Several SAS procedures, such as SGPLOT, SGSCATTER and SGPANEL, utilize the Graphic Template Language (GTL) which is a part of the Output Destination Style (ODS) Graphics software. Although these procedures have procedure-driven graph templates which produce stellar graphs they may not produce the output in exactly the format that is desired. Thus, there can be times when some of the features of the procedure-driven template will need to be adjusted to produce the graph with the desired layout. With the use of GTL, the user has the capability to modify features for graphs that are based on procedure-driven templates, as well as create completely customized graphs that cannot be produced from a procedure-driven template. Additionally, GTL makes it easier to incorporate features which in the past may have been difficult to incorporate, e.g., embedding a table of data within the output area or displaying multiple graphs, whether multiple graphs are of the same graph type or different graph types (e.g., bar chart, box plot), on the same page. Furthermore, the capacity to create ODS Graphics is part of Base SAS and therefore does not require the installation of SAS/GRAPH. These are just some of the reasons to learn GTL. According to Matange (2013, p. 5) some additional reasons to learn GTL are:

- GTL provides the full set of features that you need to create graphs from the simplest scatter plots to complex diagnostics panels in a single system.
- GTL is the language used to create the templates shipped by SAS for the creation of the standard graphs from the analytical procedures. To customize one of these graphs, you will need to understand GTL.
- GTL represents the future for analytical graphics in SAS. New features are being added to GTL with every SAS maintenance and version release.

THE BASICS

To start using GTL, you first need to understand the basics. We will discuss different types of layouts and illustrate some simple graphs with some slight modifications.

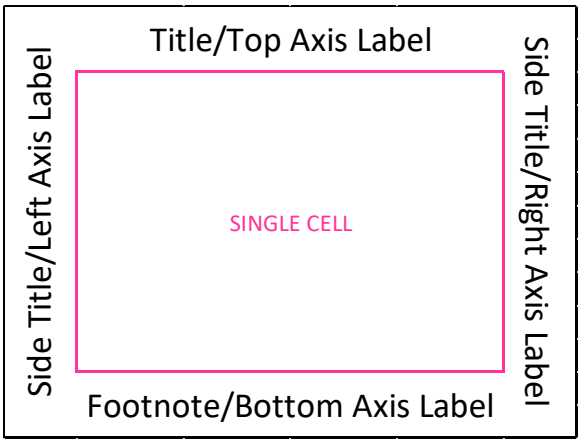
LAYOUTS

Types

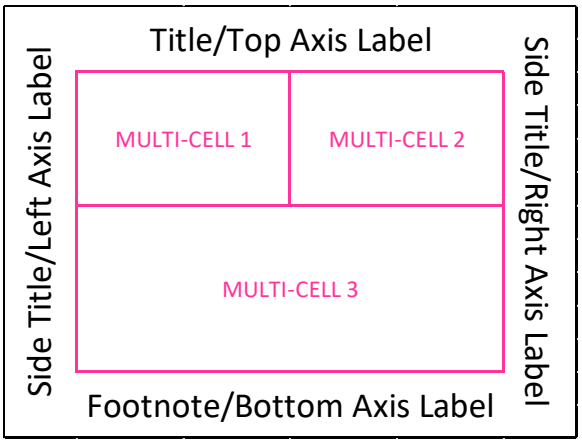
There are several types of layouts available in GTL. The type of layout is determined by what needs to be graphed. The types of layouts can be classified as single-cell, multi-cell pre-defined and multi-cell data-driven. Single-cell layout is a graph which uses the entire graphing area as illustrated in Display 1. Multi-cell pre-defined layout is a graph which breaks the graphing area into pre-defined parts so that different pieces of information can be displayed. Multi-cell data-driven layout breaks the graphing area

into as many necessary parts based on the data. Within the single-cell and multi-cell type layouts, there are various types of layouts that will help further refine what type of display is needed.

- Single-cell: A graph which uses the entire graphing area (Display 1)
    - OVERLAY: General layout with 2-D plots
    - OVERLAYEQUATED: Overlay with equated axes
    - PROTOTYPE: Specialized, used with DATAPANEL or DATALATTICE only
    - REGION: General plot with no axes
    - OVERLAY3D: General layout with 3-D plots
- Multi-cell (cells must be pre-defined): A graph that breaks the graphing area into pre-defined portions so that each portion represents different pieces of information (Display 2)
    - GRIDDED: multi-cell plots have the same proportion in regard to height and width
    - LATTICE: very flexible which allow each cell to have different heights and widths
      Both GRIDDED and LATTICE can be used along with OVERLAY. With these two layouts other types of layouts can be nested within each cell.
- Multi-cell Data-driven (2-D: Panels of similar graphs based on data classification variables): A graph that breaks the graphing area into as many parts necessary based on the data (Display 2)
    - DATAPANEL: Number of cells based on crossings of ***n*** classification variables
    - DATALATTICE: Number of cells based on crossings of 1 or 2 classification variables.
    - Both DATAPANEL and DATALATTICE need to use the PROTOTYPE layout.



Display 1. Single-cell Illustration



Display 2. Multi-cell Illustration

Understanding these various layouts and how they can be utilized to produce a graph is a key part of the use of GTL. It is also important to distinguish between the graph area and the output area. The output area is where the actual figure/table is displayed. The graph area encompasses the output area along with any titles, footnotes and axes labels. For Multi-cell layouts each cell can have its own set of axes depending on the layout used.

Sample Single-Cell Layouts

The two graphs below are both single-cell layouts. Even though the second graph (Figure 2) has two different types of graphs it is still a single-cell layout with both graphs occupying the same output area.

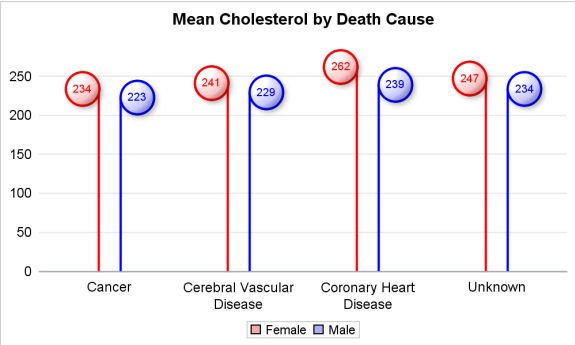


Figure 1
https://blogs.sas.com/content/graphicallyspeaking/2017/07/24/lollipop-charts/

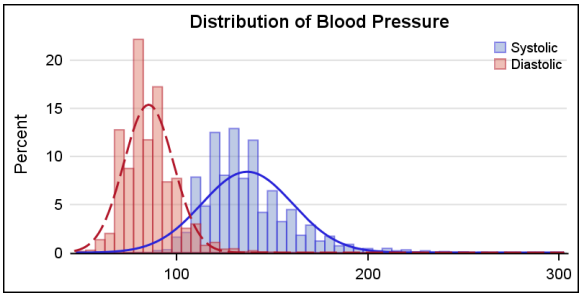


Figure 2
https://blogs.sas.com/content/graphicallyspeaking/2017/04/30/getting-started-with-sgplot-histograms/

Sample of Multi-cell Layouts

The first example of the multi-cell layout (Figure 3) utilizes the LATTICE layout and the second example of the multi-cell layout (Figure 4) utilizes the DATALATTICE layout. With the LATTICE layout we can control the size of each cell. However, with the DATALATTICE cell size is determined by the data.

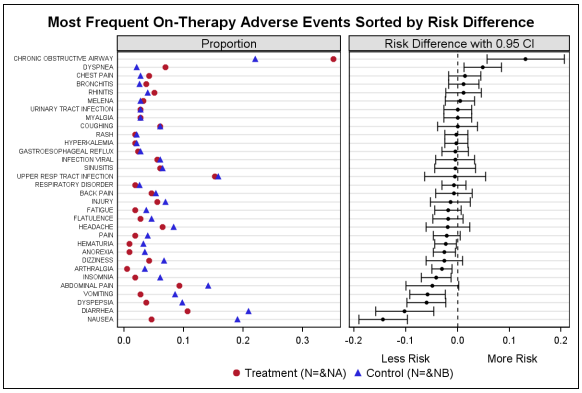


Figure 3
https://blogs.sas.com/content/graphicallyspeaking/2012/12/03/most-frequent-ae-sorted-by-relative-risk/

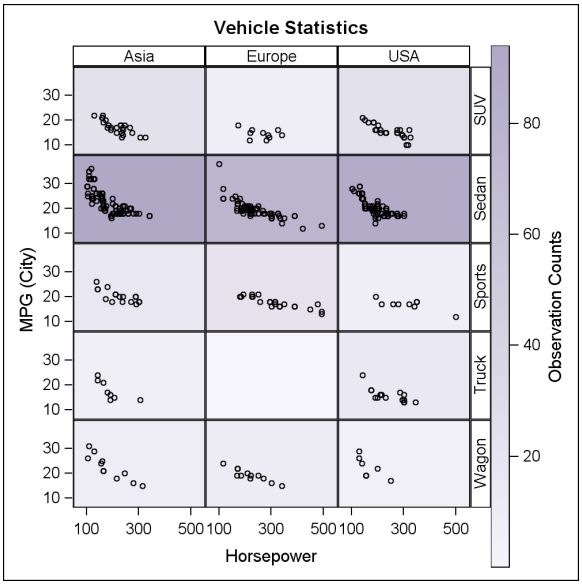


Figure 4
https://blogs.sas.com/content/graphicallyspeaking/2014/02/22/datalattice-with-gradient-backgrounds/

General Syntax

Each layout is initiated with LAYOUT and terminated with ENDLAYOUT, regardless of the type of layout selected.

The syntax for OVERLAY and GRIDDED is similar but have varying options.

```
layout type </options>;

      …  GTL Plot Statements …

      <innermargin </options>;
      block-plot or axis-table statements;
      end innermargin;>

endlayout;
```

The INNERMARGIN statement can only be used in OVERLAY and PROTOTYPE layouts.

The syntax for the LATTICE layout is also similar to OVERLAY and GRIDDED layouts but with some added components which will allow for either different axes for each portion of the graph or allow for sharing of axes. By default, LATTICE will control the axes and create the axes based on the data in the plots. However, if you need to have uniform axes (i.e., the axes need to be the same across all cells), then COLUMNDATARANGE and ROWDATARANGE options on the layout statement can be utilized. But if two sets of axes are needed, then those axes need to be defined. COLUMNAXES and COLUMN2AXES are used to control the X and X2 axes, respectively and ROWAXES and ROW2AXES are used to control the Y and Y2 axes. However, if all parts of the graph are to have the same axes, then only one set of axes statements would need to be defined.

```
layout LATTICE </options>;
      …  GTL Plot Statements …
      <columnaxes </options>;
            columnaxis / axis-option(s);
            …
      endcolumnaxes;>
      <column2axes </options>;
         columnaxis / axis-option(s);
            …
      endcolumn2axes;>
      <rowaxes </options>;
         rowaxis / axis-option(s);
            …
      endrowaxes;>
      <row2axes </options>;
         rowaxis / axis-option(s);
            …
      endrow2axes;>
      <columnheaders;
            …
      endcolumnheaders;>
      <sidebar </options>;
      endsidebar;>
endlayout;
```

There are a variety of options that can be specified that will control the look of the output. Table 1 provides a helpful table showing the options for GRIDDED, LATTICE and OVERLAY layouts, which allows you to compare the options between three layouts. Options for the other types of layouts can be found at https://documentation.sas.com/?docsetId=grstatgraph&docsetTarget=titlepage.htm&docsetVersion=9.4&locale=en under Part 3 Layout Statements.

		GRIDDED	LATTICE	OVERLAY
APPEARANCE				
ASPECTRATIO	AUTO | *number > 0*			✓
BACKGROUNDCOLOR	*style-reference color*	✓	✓	✓
BORDER	TRUE | FALSE	✓	✓	✓
BORDERATTRS	*style-element* | *style-element (line-options)* | *(line-options)*	✓	✓	✓
COLUMNGUTTER	*dimension*	✓	✓	
CYCLEATTRS	TRUE | FALSE			✓
OPAQUE	TRUE | FALSE	✓	✓	✓
OUTERPAD	AUTO | *dimension* | *(pad-options)*	✓	✓	✓
PAD	*dimension* | *(pad-options)*	✓	✓	✓
SHRINKFONTS	TRUE | FALSE	✓	✓	
WALLCOLOR	*style-reference color*			✓
WALLDISPLAY	STANDARD | ALL | NONE | *(display-options)*			✓
AXIS				
X2AXISOPTS	*axis-options*			✓
XAXISOPTS	*axis-options*			✓
Y2AXISOPTS	*axis-options*			✓
YAXISOPTS	*axis-options*			✓
CELL				
COLUMNWEIGHTS	UNIFORM | PREFERRED | *(weight-list)*		✓	
ROWWEIGHTS	UNIFORM | PREFERRED | *(weight-list)*		✓	
COLUMN				
COLUMN2DATARANGE	DATA | UNIONALL | UNION		✓	
COLUMNDATARANGE	DATA | UNIONALL | UNION		✓	
GRID				
COLUMNS	*integer*	✓		
ROWS	*integer*	✓		
LATTICE				
COLUMNS	*integer*		✓	
ROWS	*integer*		✓	
SKIPEMPTYCELLS	TRUE | FALSE		✓	
LAYOUT				
ROWGUTTER	*dimension*	✓	✓	
LEGEND				
LOCATION	INSIDE | OUTSIDE	✓		
LOCATION				

		GRIDDED	LATTICE	OVERLAY
AUTOALIGN	NONE | AUTO | *(location list)*	✓	✓	
HALIGN	CENTER | LEFT | RIGHT | *number*	✓	✓	
VALIGN	CENTER | TOP | BOTTOM | *number*	✓	✓	
PANEL				
ORDER	ROWMAJOR | COLUMNMAJOR	✓	✓	
ROW				
ROW2DATARANGE	DATA | UNIONALL | UNION		✓	
ROWDATARANGE	DATA | UNIONALL | UNION		✓	

Table 1. Options for GRIDDED, LATTICE and OVERLAY layouts

PLOTS

There are a variety of plots to choose from. Each plot will have its own syntax (Table 2) and own set of options (Table 3) available to it. Table 3 is provided to help compare a few of the different options associated with HIGHLOWPLOT, SCATTERPLOT, SERIESPLOT.

PLOT	SYNTAX
HIGHLOWPLOT	HIGHLOWPLOT X = *column* | *expression* LOW = *numeric-column* | *expression* HIGH = *numeric-column* | *expression </option(s)>;* HIGHLOWPLOT Y = *column* | *expression* LOW = *numeric-column* | *expression* HIGH = *numeric-column* | *expression </option(s)>;*
SCATTERPLOT	SCATTERPLOT X = *column* | *expression* Y = *column* | *expression </option(s)>;*
SERIESPLOT	SERIESPLOT X = *column* | *expression* Y = *column* | *expression </option(s)>;*

Table 2. Syntax for Select Plots

OPTION	ALLOWED VALUES	HIGHLOWPLOT	SCATTERPLOT	SERIESPLOT
APPEARANCE				
DISPLAY	STANDARD | ALL | *(display-options)*	✓		✓
HIGHCAP	*column* | NONE | SERIF | BARBEDARROW | FILLEDARROW | OPENARROW | CLOSEDARROW	✓		
LINEATTRS	*style-element* | *style-element (line-options)* | *(line-options)*	✓		✓
LOWCAP	*column* | NONE | SERIF | BARBEDARROW | FILLEDARROW | OPENARROW | CLOSEDARROW	✓		
MARKERATTRS	*style-element* | *style-element (marker-options)* | *(marker-options)*		✓	✓
TYPE	LINE | BAR	✓		
AXIS				
XAXIS	X | X2	✓	✓	✓
YAXIS	Y | Y2	✓	✓	✓
MIDPOINT / GROUPING				
GROUP	*column* | *discrete-attr-var* | *expression*	✓	✓	✓

Table 3. Sample of Options Available for Select Plots

The plot statements shown in Table 2 are the ones that will be illustrated in the step-by-step approach to creating a template. For a complete list of plots and all features available visit Plot Statements → Plot Statements at https://documentation.sas.com/?docsetId=grstatgraph&docsetTarget=titlepage.htm&docsetVersion=9.4&locale=en.

USING GTL TO CREATE A GRAPH

Matange (2013, p. 11) points out that using GTL to create a graph is a two-step process:

1. First you need to define the structure of the graph using the STATGRAPH template. In the creation of the template, no graph is actually produced.
2. Secondly you need to associate the data with the template produced in step 1 in order to render the template which will produce the graph.

1. DEFINE THE TEMPLATE

All templates will have the following structure.

```
proc template;
 define statgraph templatename;
    begingraph / <options>;
         layout type / <options>;

            … GTL Plot Statements …

         endlayout;
    endgraph;
 end;
run;
```

(4) (3) (2) (1)

(1) The first step to creating a custom template is to define the structure of the graph. This is done using

STATGRAPH. Each custom template will start with 'define statgraph ***templatename***' which allows the user to provide a name for the custom template. The template name is used when rendering the graph. This statement has a corresponding END.

2. Each STATGRAPH, custom template definition, has one BEGINGRAPH, which is the signal that indicates the various components of the custom template are specified within the block. BEGINGRAPH has a corresponding ENDGRAPH, which signals the end of the graph template definition.

3. Within each custom template, the layout(s) are specified along with the necessary options using the LAYOUT statement. Layouts can be nested depending on whether LATTICE, GRIDDED, DATAPANEL or DATALATTICE is used. For each LAYOUT, you need to signal the end of the layout with ENDLAYOUT.

4. Most of the work will take place within the LAYOUT block(s). Depending on the type of layout(s) and plot(s) will drive how this portion is programmed. If there are multiple layouts or nested layouts, then there needs to be a corresponding ENDLAYOUT for each. As many plot statements and their corresponding options needed to build the graph should be encompassed within the corresponding LAYOUT.

2. PRODUCE THE GRAPH

As Matange indicates, using GTL is a two-step process. In the second step, SGRENDER is used to associate the data that will be used with the custom template that is defined in step 1. Any data can be associated with the template if all the components (i.e., variables) defined in the template reside in the data set. Syntax for SGRENDER

```
proc sgrender data = datasetname template = templatename;
   <optional SAS statements>;
run;
```

Some examples of other SAS statements that can be incorporated are:

- BY statement to allow rendering by different groups.
- FORMAT statement to allow data to be formatted without losing the order of the data.
- LABEL statement to allow x and y-axis labels to be defined if they are not defined in the template.

EXAMPLE OF CREATING THE IMPOSSIBLE

Up to this point, you have seen the basics for GTL. How does it all work together to produce something like a patient profile plot as illustrated in Figure 5 below?

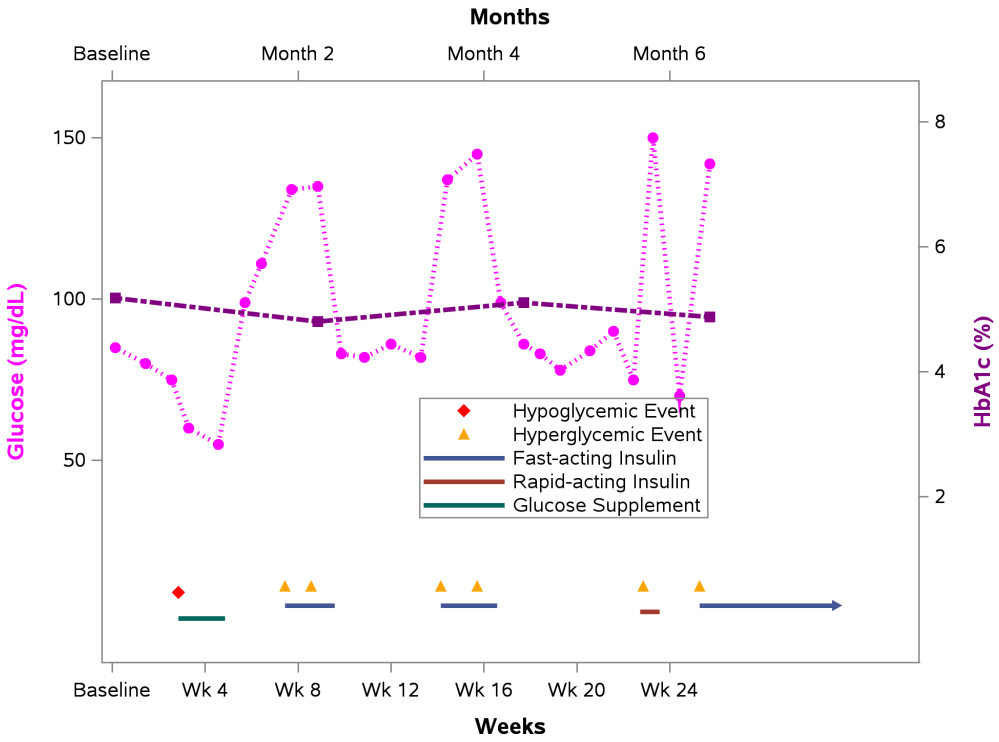


Figure 5. Patient 1 Profile of Glucose and HbA1C Over Time Along with Hyper/Hypoglycemic Events and Rescue Medication

Patient profile plots provide a lot of information in a graphic representation. We would like to get two lab results that are captured at different schedules on the same plot along with each episode of a particular event, and in addition indicate when they took a rescue medication.

At first glance, such a plot may seem very overwhelming and near impossible to do. If you break it down into the various pieces and start with what you know how to do, it then becomes manageable.

EXAMPLE SAS PROGRAM 1

The beginning points for this graphic are two series plots which have to be constructed before even considering how to incorporate the events and medications. Let's examine what the data looks like before determining how to set up our template. Data Display 1 shows that Glucose levels are determined approximately every week, while the HbA1c levels are determined every two months. The two data sources have different time intervals and this needs to be accounted for when building the graph.

Starting with the first lab test, you can produce a series plot as shown in Figure 6 using SAS Program 1.

USUBJID	ADY	ADT	HBA1C	SGLUC
ABC-DEF-0001	1	5-Nov-16	5.18	85
ABC-DEF-0001	10	14-Nov-16		80
ABC-DEF-0001	18	22-Nov-16		75
ABC-DEF-0001	23	27-Nov-16		60
ABC-DEF-0001	32	6-Dec-16		55
ABC-DEF-0001	40	14-Dec-16		99
ABC-DEF-0001	45	19-Dec-16		111
ABC-DEF-0001	54	28-Dec-16		134
ABC-DEF-0001	62	5-Jan-17	4.8	135
ABC-DEF-0001	69	12-Jan-17		83
ABC-DEF-0001	76	19-Jan-17		82
ABC-DEF-0001	84	27-Jan-17		86

Data Display 1. Sample Glucose and HbA1c data

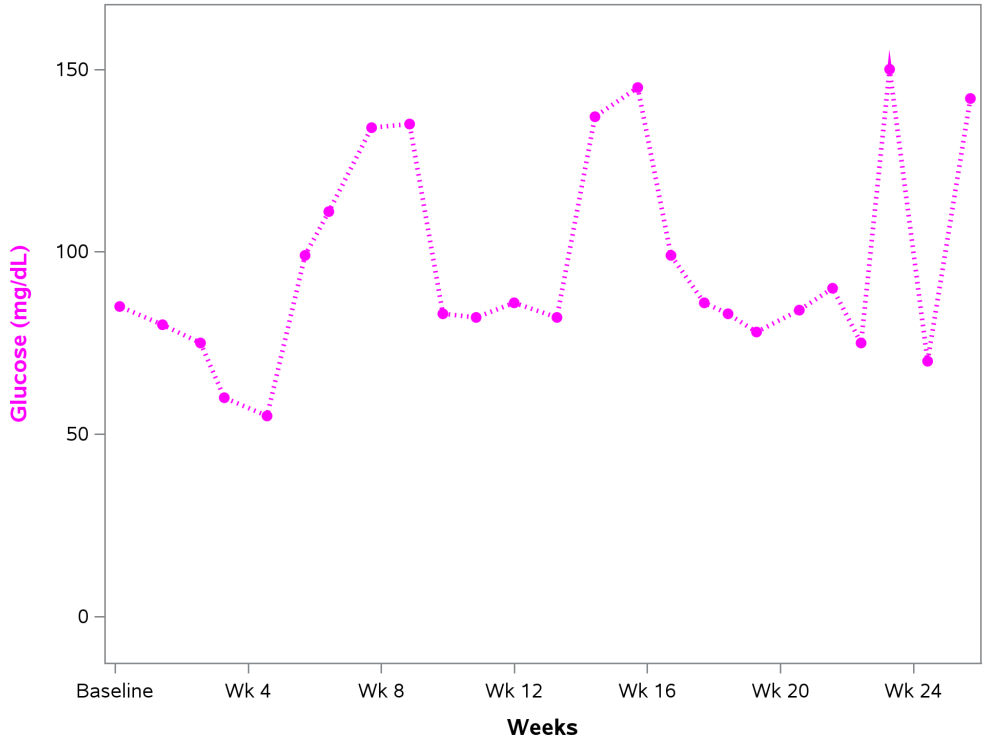


Figure 6. Series Plot of Glucose Results Over Time by Week

```
proc template;
   define statgraph glucprof;
      begingraph / border = false;
         layout overlay / xaxisopts = (label = 'Weeks'
                                       labelattrs = (color = black weight = bold)
                                       linearopts = (viewmin = 0 viewmax = 180
                                                     tickvaluesequence = (start = 0
                                                                          end = 168
                                                                          increment = 28)
                                                     tickvalueformat = dy2wk.))
                          yaxisopts = (label = 'Glucose (mg/dL)'
                                       offsetmin = 0.07 offsetmax = 0.07
                                       labelattrs = (color = fuchsia weight = bold)
                                       linearopts = (viewmin = 0 viewmax = 155));

           seriesplot x = STRTDAY y = SGLUC / display = all
                                              markerattrs = (symbol = circlefilled
                                                             color = fuchsia size = 6)
                                              lineattrs = graphdata3(color = fuchsia
                                                                     pattern = 34
                                                                     thickness = 3);

         endlayout;
      endgraph;
   end;
run;
```

SAS Program 1. PROC TEMPLATE to Produce Series Plot of Glucose Results Over Time by Week

(1) Within each type of layout there are various options. In this example, the X-axis is indicated as having a label of 'Weeks' and the Y-axis is indicated as having a label of 'Glucose (mg/dL)'. Both axes will have the data type automatically detected. Type could be specified, in this case of the x-axis, it would be linear. In addition, to specifying the label, attributes can also be specified. You can specify label attributes such as: color, weight (i.e., normal or bold), font family, size and style (i.e., normal or italic).

(2) There are also type specific options that can be specified. Since the data type is linear, the LINEAROPTS is used. For the X-axis, a label is provided and TICKSEQUENCE is used to specify a START and END value for the X-axis. By providing the start and end value it forces the X-axis to go from 0 to 168 rather than letting it default to the maximum value. In addition, the INCREMENT indicated that only tick marks and tick values should be every 28 (days). Also, the VIEWMIN and VIEWMAX is used to indicate that only data within that range should be displayed. Since we know the data only goes to 168, the VIEWMIN and VIEWMAX could be eliminated but if you have data that may have outliers that you do not want to display, then you can cut the data off at the minimum and maximum value to display. This does not change the data it just changes what was displayed. With the use of the linear option TICKVALUEFORMAT, this allows you to control how a specific value is displayed on an axis by specifying a format.

(3) OFFSETMIN indicates how much space to reserve from the bottom axis. OFFSETMAX indicates how much space to reserve from top axis. If OFFSETMIN is used on xaxisopts then it is the space reserved from the left and OFFSETMAX is the space reserved from the right.

(4) The SERIESPLOT will have its own set of options that can be used to dictate how it should look. For this example, we want to display the series line along with the markers, so we need to use DISPLAY = ALL. The default is DISPLAY = STANDARD which will only display the series line without the markers.

(5) Markers to represent each observation can be associated attributes that control the appearance. The marker is the symbol that is used to represent each point on the series plot. Within the MARKERATTRS option there are various other options that can be used. The marker options are enclosed within parentheses to indicate that these options pertain to the marker. As shown in Figure 6 the marker used is a fuchsia filled circle that has a size of 6.

6 Similar to markers, the line that connects the observations can have attributes that are assigned that allow for customization of the display. GRAPHDATA3 is a pre-defined style element that uses default values for marker, marker size, line style, line thickness, marker and line colors. These features can be overridden by indicating what you want to override. In this example, we are using a fuchsia dotted line (pattern = 34) with a thickness of 3 to connect the fuchsia filled circles which have a size of 6. Note it is not necessary to specify GRAPHDATA3 if you are overriding all features used for the graph as is done in this example. It is only done here for illustration purpose.

EXAMPLE SAS PROGRAM 2

It is simple enough to do an individual SERIESPLOT but how do we get both series plots on the same graph? The use of the X2 and Y2 axes will allow for the display of additional results as illustrated in Figure 7. You will see that the Glucose results uses the left and bottom axes while HbA1c results uses the right and top axes (refer to SAS Program 2).

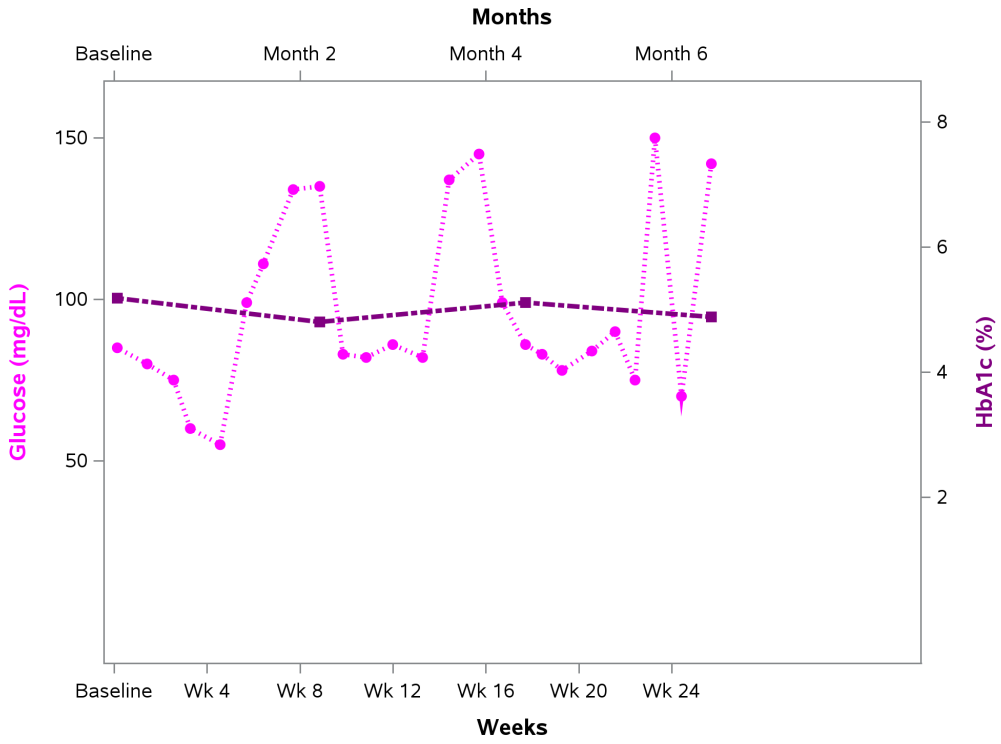


Figure 7. Series Plot of Glucose and HbA1C Results Over Time by Week

```
proc template;
   define statgraph glhbprof;
      begingraph / border = false;
         layout overlay / xaxisopts = (label = 'Weeks'
                                       labelattrs = (color = black weight = bold)
                                       linearopts = (viewmin = 0 viewmax = 240
                                                     tickvaluesequence = (start = 0
                                                                          end = 168
                                                                          increment = 28)
                                                     tickvalueformat = dy2wk.))
                          yaxisopts = (label = 'Glucose (mg/dL)'
                                       offsetmin = 0.07 offsetmax = 0.07
                                       labelattrs = (color = fuchsia weight = bold)
                                       linearopts = (viewmin = 0 viewmax = 155
                                                     tickvaluesequence = (start = 50
                                                                          end = 150
                                                                          increment = 50)))

                          x2axisopts = (label = 'Months'
                                        labelattrs = (color = black weight = bold)
                                        linearopts = (viewmin = 0 viewmax = 240
                                                      tickvaluelist = (0 56 112 168)
                                                      tickvalueformat = dy2mo.))

                          y2axisopts = (label = 'HbA1c (%)'
                                        offsetmin = 0.07
                                        offsetmax = 0.07
                                        labelattrs = (color = purple weight = bold)
                                        linearopts = (viewmin = 0 viewmax = 8
                                                      tickvaluesequence = (start = 2
                                                                           end = 8
                                                                           increment = 2)));
            /* produce portion for Serum Glucose */
            seriesplot x = STRTDAY y = SGLUC / display = all
                                               markerattrs = (symbol = circlefilled
                                                              color = fuchsia size = 6)
                                               lineattrs = graphdata3(color = fuchsia
                                                                      pattern = 34
                                                                      thickness = 3);

            /* produce portion for HbA1c */
            seriesplot x = STRTDAY y = HBA1C / xaxis = x2
                                               yaxis = y2
                                               display = all
                                               markerattrs = (symbol = squarefilled
                                                              color = purple size = 6)
                                               lineattrs = graphdata3(color = purple
                                                                      pattern = 12
                                                                      thickness = 3);
         endlayout;
   endgraph;
end;
run;
```

SAS Program 2. PROC TEMPLATE to Produce Series Plot of Glucose and HbA1C Results Over Time

(1) When using the right and top axes, the X2AXISOPTS and Y2AXISOPTS need to be specified. Options available for XAXISOPTS and YAXISOPTS are available for the X2AXISOPTS and Y2AXISOPTS as well.

(2) If you want only specific values to appear as tick marks on the axis, then you can use the TICKVALUELIST to indicate what values should be displayed.

(3) When dealing with multiple axes, you need to specify axes the plot will be using. Otherwise it will default to X-AXIS and Y-AXIS. The HbA1c (%) results were required to be on the right-hand side of the plot, and therefore yaxis=y2 was used. In addition, xaxis=x2 was used so that the results could be seen every two months across the top of the graph instead of the bottom of the graph.

EXAMPLE SAS PROGRAM 3

Now that we have the two series plots, it is time to tackle adding in the events. Again, we first need to examine the data (Data Display 2). Adding in the events can be done using SCATTERPLOT since we are only adding a marker at a specific time point of when the event occurred as see in Figure 8. Note that for the purpose of building this graph, each type of event (hypoglycemic and hyperglycemic) were assigned a numeric value (Hypoglycemic = 9 and Hyperglycemic = 11). However, to get them to appear with different markers and different colors, each event had to be created as its own numeric variable. Although no new option/concept is introduced with this portion of the graph, the SAS code is included for reference (refer to SAS Program 3).

USUBJID	AEDECOD	ASTDT	STRTDAY	EVENT1	EVENT2
ABC-DEF-0001	Hyperglycemia	30-Apr-17	177		11
ABC-DEF-0001	Hyperglycemia	13-Apr-17	160		11
ABC-DEF-0001	Hyperglycemia	22-Feb-17	110		11
ABC-DEF-0001	Hyperglycemia	11-Feb-17	99		11
ABC-DEF-0001	Hyperglycemia	3-Jan-17	60		11
ABC-DEF-0001	Hyperglycemia	26-Dec-16	52		11
ABC-DEF-0001	Hypoglycemia	24-Nov-16	20	9	

Data Display 2. Sample Hypoglycemic and Hyperglycemic Events Data

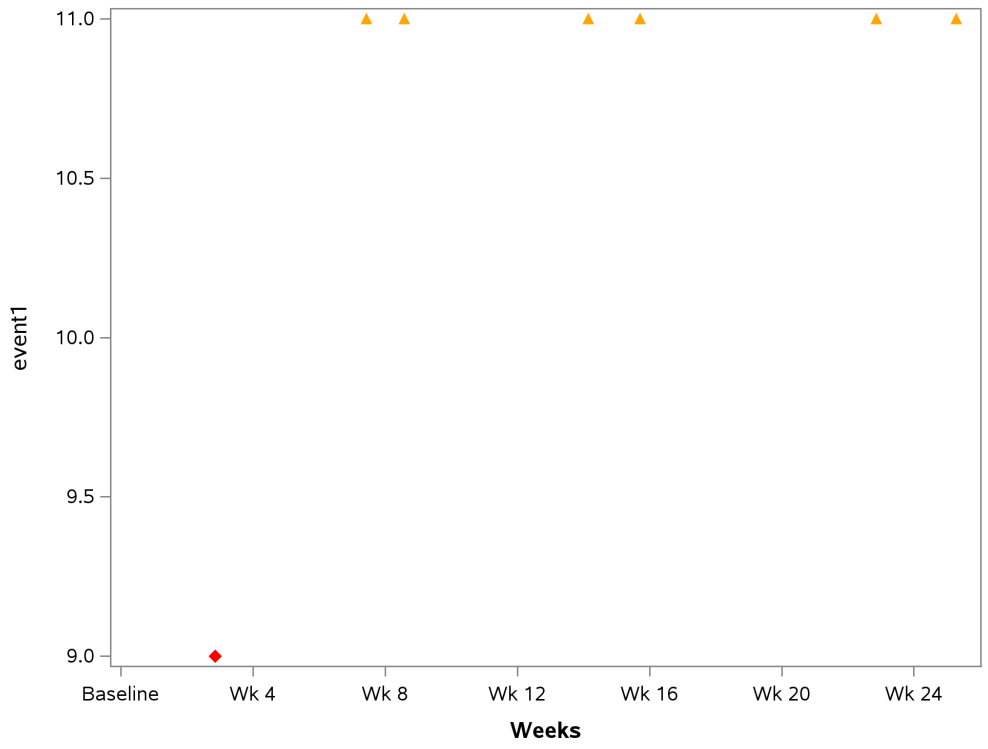


Figure 8. Scatter Plot of Hypoglycemic and Hyperglycemic Over Time by Week

```
 proc template;
    define statgraph aeprof;
       begingraph / border = false;
          layout overlay / xaxisopts = (label = 'Weeks'
                                        labelattrs = (color = black weight = bold)
                                        linearopts = (viewmin = 0 viewmax = 180
                                                      tickvaluesequence = (start = 0
                                                                           end = 168
                                                                           increment = 28)
                                                      tickvalueformat = dy2wk.));

             scatterplot x = STRTDAY y = EVENT1 / markerattrs = (symbol = diamondfilled
                                                                 color = red size = 6);

             scatterplot x = STRTDAY y = EVENT2 / markerattrs = (symbol = trianglefilled
                                                                 color = orange size =6);

          endlayout;
    endgraph;
 end;
run;
```

SAS Program 3. PROC TEMPLATE to Produce Scatter Plot of Hypoglycemic and Hyperglycemic Events

EXAMPLE SAS PROGRAM 4

Adding in the medications (Data Display 3) is handled differently than the events. With the events, we used a SCATTERPLOT since the event was at a specific point in time. However, medications can be taken over a period of time and the duration of use needed to be displayed as illustrated in Figure 9. Similar to the events, the medications were assigned a numeric value for purpose of the graph (Glucose Supplement = 1, Rapid-acting Insulin = 3, Fast-Acting Insulin = 5, Elevating Agent = 7). SAS Program 4 demonstrates how a HIGHLOWPLOT can be created.

Row	USUBJID	CMDECOD	ACAT	ASTDT	AENDT
1	ABC-DEF-0001	FlexPen	Fast-acting	30-Apr-17	.
2	ABC-DEF-0001	NovoLog	Rapid-acting	12-Apr-17	18-Apr-17
3	ABC-DEF-0001	Apidra	Fast-acting	11-Feb-17	28-Feb-17
4	ABC-DEF-0001	NovoLog	Fast-acting	26-Dec-16	10-Jan-17
5	ABC-DEF-0001	Dextrose	Glucose Supplement	24-Nov-16	8-Dec-16
6	ABC-DEF-0002	Glucagon	Elevating Agent	22-Jan-17	.
7	ABC-DEF-0002	FlexPen	Rapid-acting	14-Dec-16	9-Jan-17
8	ABC-DEF-0002	Dextrose	Glucose Supplement	10-Oct-16	11-Nov-16

Row	CMENRTPT	STRTDAY	CMCAP	ENDDAY	MED
1	ONGOING	177	FILLEDARROW	220	5
2		159	NONE	165	3
3		99	NONE	116	5
4		52	NONE	67	5
5		20	NONE	34	1
6	ONGOING	161	FILLEDARROW	185	7
7		122	NONE	148	3
8		57	NONE	89	1

Data Display 3. Sample Rescue Medication Data

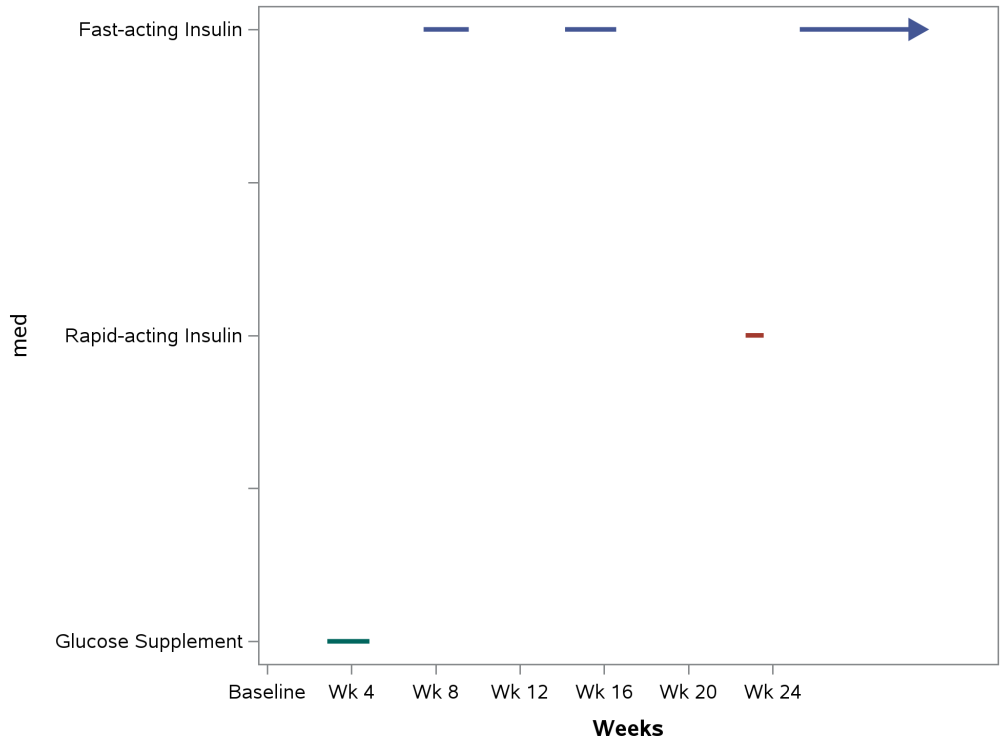


Figure 9. High-low Plot of Medications Over Time

```
proc template;
   define statgraph medprof;
      begingraph / border = false;
         layout overlay / xaxisopts = (label = 'Weeks'
                                       labelattrs = (color = black weight = bold)
                                       linearopts = (viewmin = 0 viewmax = 240
                                                     tickvaluesequence = (start = 0
                                                                          end = 168
                                                                          increment = 28)
                                                     tickvalueformat = dy2wk.));

          highlowplot y = MED low = STRTDAY high = ENDDAY / group = MED
                                                            lineattrs = (pattern=solid
                                                                         thickness=3)
                                                            type = line
                                                            highcap = cmcap;

         endlayout;
   endgraph;
end;
run;
```

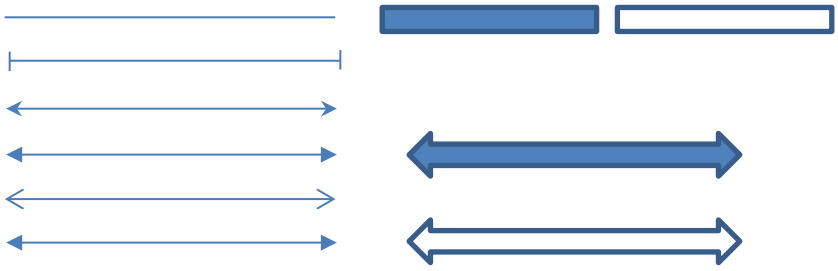
(1) (2)

SAS Program 4. PROC TEMPLATE to Produce High-low Plot of Medications Over Time

(1) The HIGHLOWPLOT will connect the minimum (LOW=) and maximum (HIGH=) value for each categorical value to create a 'floating' line or bar. If the HIGHLOWPLOT is to be displayed horizontally then the categorical values are specified with Y=. If the HIGHLOWPLOT is to be displayed vertically then the categorical values are specified with X=. GROUP creates HIGHLOWPLOT for each group, i.e., each level of medication. By default, each unique group will be represented by a different color.

2 With each floating bar, the start and end have a cap. How the cap is displayed is based on the value of LOWCAP or HIGHCAP. LOWCAP indicates the type of cap at the low end of the 'floating' bar, while HIGHCAP indicates the type for the high end of the bar. The possible types of cap are:

- NONE
- SERIF
- BARBEDARROW
- FILLEDARROW
- OPENARROW
- CLOSEDARROW



If the option is not specified, then the default value is NONE. In our example, the values of the cap are stored in a variable name CMCAP. This allows for the HIGHCAP to vary from observation to observation. As illustrated in Figure 9 one of the end caps is FILLEDARROW while the others are NONE.

EXAMPLE SAS PROGRAM 5

With all the components programmed, they can be combined to make the final graph with some minor additions.

```
proc template;
   define statgraph ptprof;
      begingraph / border = false;
         layout overlay / xaxisopts = (label = 'Weeks'
                                       labelattrs = (color = black weight = bold)
                                       linearopts = (viewmin = 0 viewmax = 240
                                                     tickvaluesequence = (start = 0
                                                                          end = 168
                                                                          increment = 28)
                                                     tickvalueformat = dy2wk.))
                          yaxisopts = (label = 'Glucose (mg/dL)'
                                       offsetmin = 0.07 offsetmax = 0.07
                                       labelattrs = (color = fuchsia weight = bold)
                                       linearopts = (viewmin = 0 viewmax = 155
                                                     tickvaluesequence = (start = 50
                                                                          end = 150
                                                                          increment = 50)))
                          x2axisopts = (label = 'Months'
                                       labelattrs = (color = black weight = bold)
                                       linearopts = (viewmin = 0 viewmax = 240
                                                     tickvaluelist = (0 56 112 168)
                                                     tickvalueformat = dy2mo.))
                          y2axisopts = (label = 'HbA1c (%)'
                                       offsetmin = 0.07 offsetmax = 0.07
                                       labelattrs = (color = purple weight = bold)
                                       linearopts = (viewmin = 0 viewmax = 8
                                                     tickvaluesequence = (start = 2
                                                                          end = 8
                                                                          increment = 2)));

            /* produce portion for Serum Glucose */
            seriesplot x = STRTDAY y = SGLUC / display = all
                                               markerattrs = (symbol = circlefilled
                                                              color = fuchsia size = 6)
                                               lineattrs = graphdata3(color = fuchsia
                                                                      pattern = 34
                                                                      thickness = 3);
```

```
            /* produce portion for HbA1c */
            seriesplot x = STRTDAY y = HBA1C / xaxis = x2
                                               yaxis = y2
                                               display = all
                                               markerattrs = (symbol = squarefilled
                                                              color = purple size = 6)
                                               lineattrs = graphdata3(color = purple
                                                                      pattern = 12
                                                                      thickness = 3);

            /* produce portion for hypoglycemic events */
            scatterplot x = STRTDAY y = EVENT1 / markerattrs = (symbol = diamondfilled
                                                                color = red size = 6)
                                                 name = 'hypo'
                                                 legendlabel = 'Hypoglycemic Event';

            /* produce portion for hyperglycemic events */
            scatterplot x = STRTDAY y = EVENT2 / markerattrs = (symbol = trianglefilled
                                                                color = orange size = 6)
                                                 name = 'hyper'
                                                 legendlabel = 'Hyperglycemic Event';

            /* produce portion for medication */
            highlowplot y = MED low = STRTDAY high = ENDDAY / group = MED
                                                              lineattrs = (pattern=solid
                                                                           thickness=3)
                                                              type = line
                                                              display = all
                                                              highcap = cmcap
                                                              name = 'med';

            /* create a legend that will explain the symbols for events and meds     */
            /* legend location will vary based on available space but will be inside */
            /* need to exclude any part of the legend that represents missing values */
            discretelegend 'hypo' 'hyper' 'med' / location = inside
                                                  across = 1
                                                  autoalign = auto
                                                  exclude = (" " ".");
        endlayout;
    endgraph;
 end;
run;
```

SAS Program 5. PROC TEMPLATE to Produce Patient Profile of Glucose and HbA1C Over Time Along with Hyper/Hypoglycemic Events and Rescue Medication

(1) The NAME option allows the plot statement to be referenced in other PROC TEMPLATE statements such as when defining the legend. The LEGENDLABEL option indicates what label should be displayed for that particular plot in the DISCRETELEGEND. Note that for the HIGHLOWPLOT there is no LEGENDLABEL provided and that is because the labels will come from the data values.

(2) DISCRETELEGEND will create a legend that relates to the various plots in the graph. For the SCATTERPLOTs and the HIGHLOWPLOT we provided a 'name' for each plot so that it could be referenced elsewhere in the PROC TEMPLATE. The 'name' allows us to reference the plot within the DISCRETELEGEND and create a legend for each. For example, 'hypo' creates the legend for hypoglycemic markers, 'hyper' creates the legend for hyperglycemic markers and 'med' creates the legend for the different types of rescue medications. Within the DISCRETELEGEND there are several types of options that allow you to control how it is displayed and where it is displayed. In this example, we want to display the legend with one value across on the inside of the graph but we want SAS to figure out where it will best fit since for each patient because it may not always fit in the same location as shown in Figure 10 and Figure 11. The EXCLUDE option indicates which entries should not be included in the legend. In situations such as this where some values may be missing due to a patient either not having

an event and/or needing rescue medication we do not want a blank value appearing for these items in the legend.

Once the final template is created, then the data needs to be associated with the template using PROC SGRENDER. In order to associate the data all data must reside in the same data set, so all three types of data can be set together into one master data set. Although the variables may not align, the template will recognize when there are null values for some of the records and will bypass those records and only process the records with data associated with the specific portion of the template. This yields the final output for patient 1 as illustrated in Figure 10.

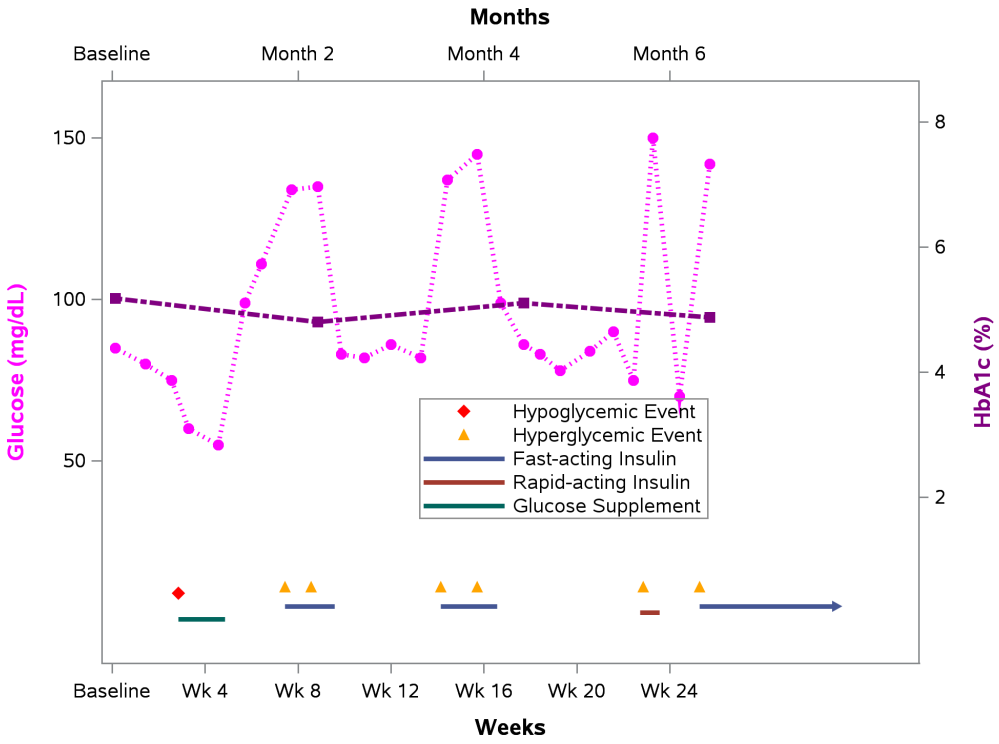


Figure 10 Final Patient 1 Profile Plot

For each patient, the legend with the specific options specified is included and the location of the legend is dependent on the data as evident in comparison of the legends in Figure 10 and Figure 11.

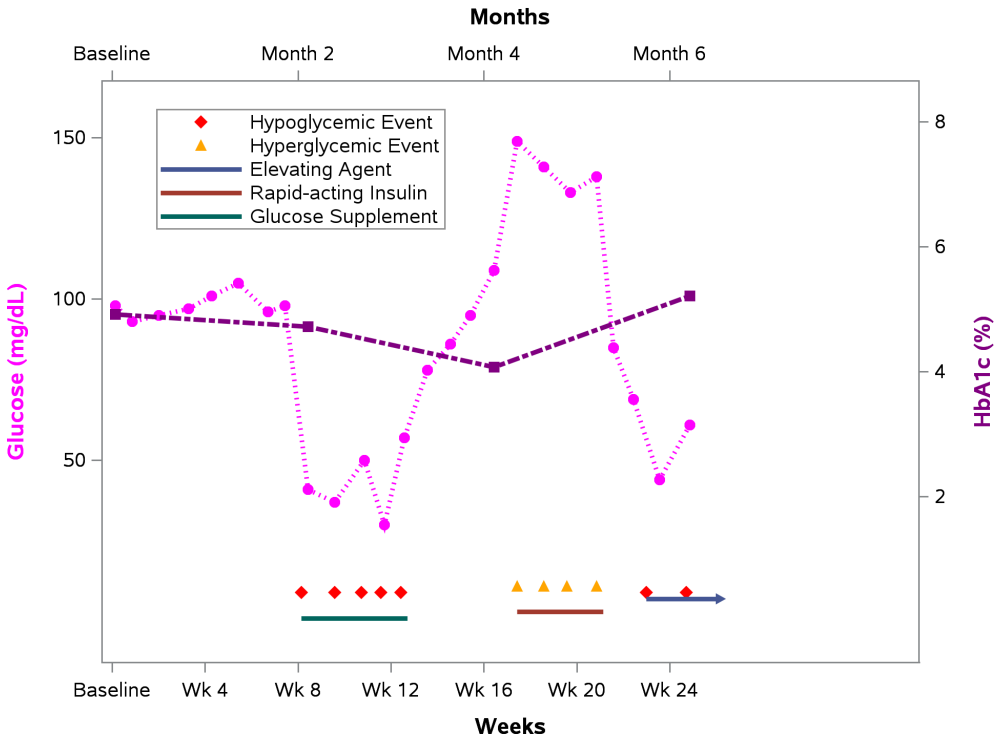


Figure 11. Patient 2 Profile Plot with Legend in a Different Location within the Graph

CONTROLLING LEGEND COLORS

In some cases, not all patients will have the same data available, so how do we ensure that the legend is consistent across subjects? As we have seen in the two figures above the legend is slightly different between patient 1 and patient 2. Patient 2 does not have 'Fast-acting Insulin' but has 'Elevating Agent'. For patient 1 'Fast-acting Insulin' is blue but for Patient 2 'Elevating Agent' is blue. This could lead to possible confusion since blue is representing two different medications and could potentially represent other medications based on the data. But for ease of review, we want to make sure they do match. There are two approaches that can be taken to ensure they remain consistent. Both approaches use DISCRETEATTRVAR statement in the template. How the attribute map is defined is what differs.

DEFINING ATTRIBUTE MAP WITHIN PROC TEMPLATE

The first approach for defining the attribute map is defining it within PROC TEMPLATE itself.

```
proc template;
   define statgraph ptprof;
      begingraph / border = false;

         /* define the color attribute map */
         discreteattrmap name = "medcolor" / ignorecase = true trimleading = true;
            value 'Glucose Supplement' / lineattrs = (color = green);
            value 'Elevating Agent'    / lineattrs = (color = lightslategray);
            value 'Rapid-acting'       / lineattrs = (color = firebrick);
            value 'Fast-acting'        / lineattrs = (color = blue);
         enddiscreteattrmap;

         /* associate the color attribute map (medcolor) with input data (dummy3) */
         /* provide a name for the association to be referenced later (colorvar)  */
         discreteattrvar attrvar = colorvar var = ACAT attrmap = "medcolor";
```

1

2

```
        layout overlay / xaxisopts = (label = 'Weeks'
                                      labelattrs = (color = black weight = bold)
                                      linearopts = (viewmin = 0 viewmax = 240
                                                    tickvaluesequence = (start = 0
                                                                         end = 168
                                                                         increment = 28)
                                                    tickvalueformat = dy2wk.))
                         yaxisopts = (label = 'Glucose (mg/dL)'
                                      offsetmin = 0.07 offsetmax = 0.07
                                      labelattrs = (color = fuchsia weight = bold)
                                      linearopts = (viewmin = 0 viewmax = 155
                                                    tickvaluesequence = (start = 50
                                                                         end = 150
                                                                         increment = 50)))
                         x2axisopts = (label = 'Months'
                                      labelattrs = (color = black weight = bold)
                                      linearopts = (viewmin = 0 viewmax = 240
                                                    tickvaluelist = (0 56 112 168)
                                                    tickvalueformat = dy2mo.))
                         y2axisopts = (label = 'HbA1c (%)'
                                      offsetmin = 0.07 offsetmax = 0.07
                                      labelattrs = (color = purple weight = bold)
                                      linearopts = (viewmin = 0 viewmax = 8
                                                    tickvaluesequence = (start = 2
                                                                         end = 8
                                                                         increment = 2)));

           /* produce portion for Serum Glucose */
           seriesplot x = STRTDAY y = SGLUC / display = all
                                              markerattrs = (symbol = circlefilled
                                                             color = fuchsia size = 6)
                                              lineattrs = graphdata3(color = fuchsia
                                                                     pattern = 34
                                                                     thickness = 3);

           /* produce portion for HbA1c */
           seriesplot x = STRTDAY y = HBA1C / xaxis = x2
                                              yaxis = y2
                                              display = all
                                              markerattrs = (symbol = squarefilled
                                                             color = purple size = 6)
                                              lineattrs = graphdata3(color = purple
                                                                     pattern = 12
                                                                     thickness = 3);

           /* produce portion for hypoglycemic events */
           scatterplot x = STRTDAY y = EVENT1 / markerattrs = (symbol = diamondfilled
                                                               color = red size = 6)
                                                name = 'hypo'
                                                legendlabel = 'Hypoglycemic Event';

           /* produce portion for hyperglycemic events */
           scatterplot x = STRTDAY y = EVENT2 / markerattrs = (symbol = trianglefilled
                                                               color = orange size = 6)
                                                name = 'hyper'
                                                legendlabel = 'Hyperglycemic Event';


           /* produce portion for medication */
           highlowplot y = med low = strtday high = endday / group = colorvar (3)
                                                    lineattrs = (pattern = solid
                                                                 thickness = 3)
                                                    type = line
```

```
                                                    display = all
                                                    highcap = cmcap
                                                    name = 'med';

            /* create a legend that will explain the symbols for events and meds     */
            /* legend location will vary based on available space but will be inside */
            /* need to exclude any part of the legend that represents missing values */
            discretelegend 'hypo' 'hyper' 'med' / location = inside
                                                  across = 1
                                                  autoalign = auto
                                                  exclude = (" " ".");
     endlayout;
   endgraph;
 end;
run;
```

SAS Program 6 Defining Color Attribute within PROC TEMPLATE

1. The DISCRETEATTRMAP statement allows the user to specify a specific color for each formatted data value. Note that you must use the formatted value otherwise this will not yield the correct color scheme. In this statement the attribute map is provided a name. In this case we are referring to the attribute map as “medcolor”. We are indicating that regardless of the case of the values if a match is found the appropriate color is assigned. In addition, we want to remove any leading blank spaces. So, if the values have leading spaces then those are disregarded when determining if the value is one of the values defined in the attribute map. We can then define each value and the associated color and any line options such as width and line type, at least one value must be defined. If more than one value has the same set of attributes, then they can be on the same VALUE line but must be separated by a blank space and each individual value must be in its own set of quotation marks. The value statement can have OTHER (not in quotes) to create a category of all other values. If OTHER is not specified, then those data values that do not have a VALUE statements will have attributes assigned as if there were no attribute map. As with most statements in GTL there needs to be a signal to indicate the end of the attribute map. This is done with ENDDISCRETEATTRMAP. Note that DISCRETEATTRMAP **must** be placed inside the template between the BEGINGRAPH statement and the first LAYOUT statement.
2. Once the attribute map is defined, we need to associate the attribute with the associated variable and we then must provide the association a name. To do this we use the DISCRETEATTRVAR statement. In this example, we associate the “medcolor” attribute map with the ACAT variable and assign the association the name colorvar.
3. The name assigned for the association between the attribute map and the variable can then be used in the appropriate plot statement.

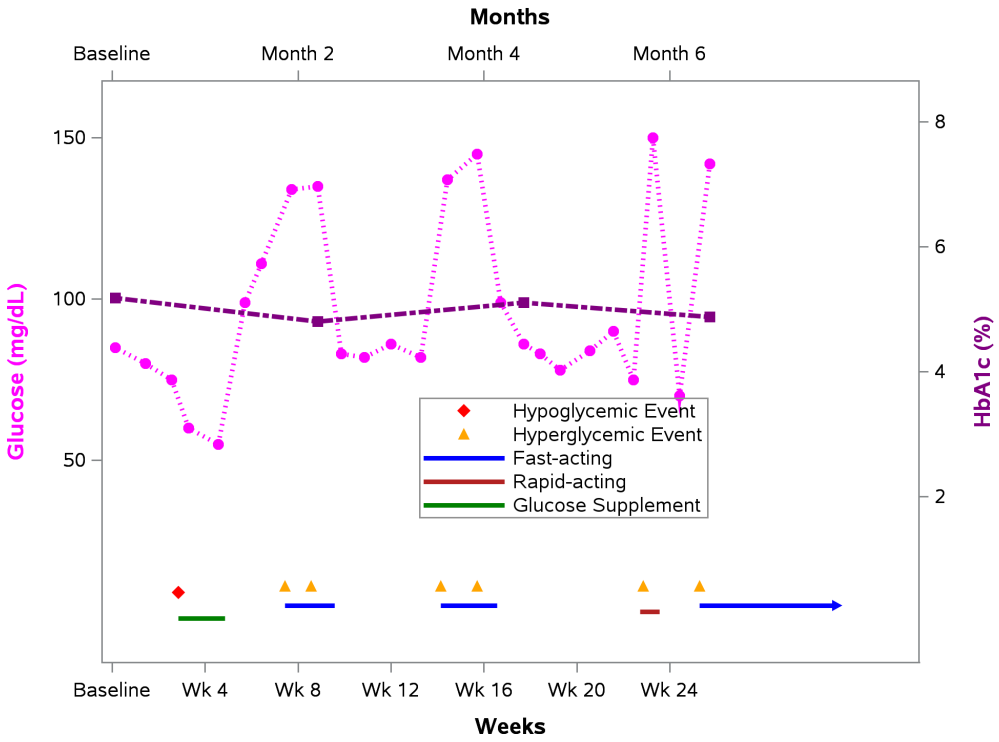


Figure 12 Patient 1 Profile Plot with DISCRETEATTRVAR

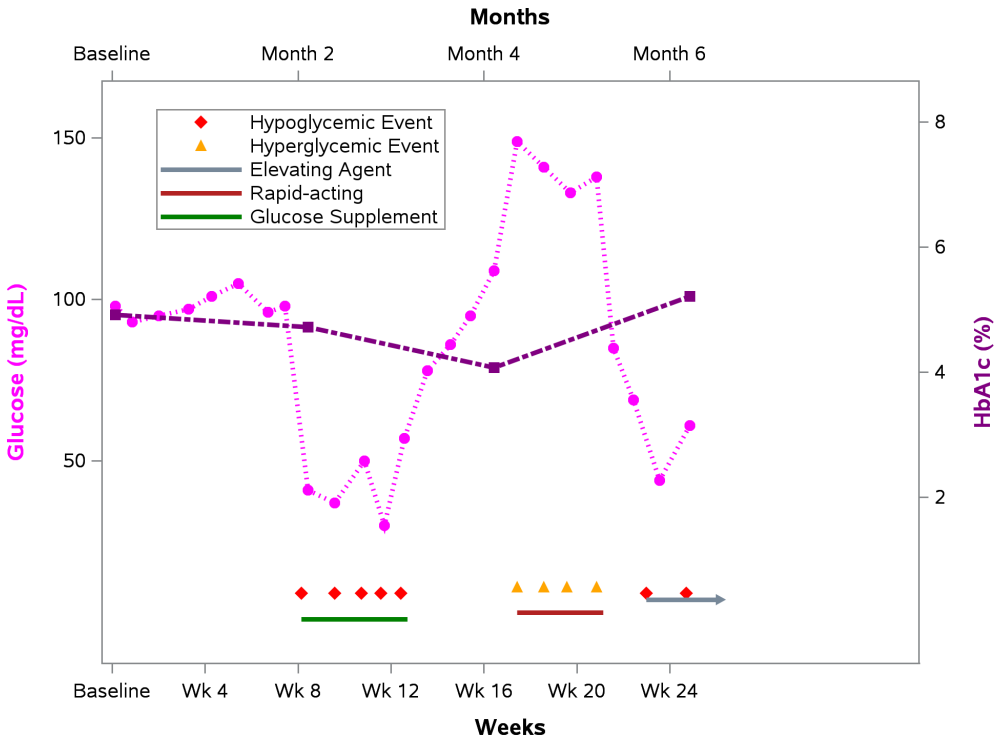


Figure 13 Patient 2 Profile Plot with DISCRETEATTRVAR

DEFINING ATTRIBUTE MAP WITHIN A DATA SET

An alternate approach for defining the attribute map is to create a data set that contains the necessary fields to specify the various attributes associated with the values. The template that is defined is the same as in SAS Program 6 with the exception that the DISCRETEATTRMAP portion of the code is removed and the information is stored in a SAS data set (SAS Program 8).

```
/* define the color attribute map */
discreteattrmap name = "medcolor" / ignorecase = true trimleading = true;
   value 'Glucose Supplement' / lineattrs = (color = green);
   value 'Elevating Agent'    / lineattrs = (color = lightslategray);
   value 'Rapid-acting'       / lineattrs = (color = firebrick);
   value 'Fast-acting'        / lineattrs = (color = blue);
enddiscreteattrmap;
```

SAS Program 7. DISCRETEATTRMAP Statement to be Removed

```
data colormaps;
   input ID $1-8 VALUE $10-27 LINECOLOR $29-42;
   datalines;
medcolor Glucose Supplement green
medcolor Elevating Agent    lightslategray
medcolor Rapid-acting       firebrick
medcolor Fast-acting        blue
;
run;
```

SAS Program 8. Attribute Mapping Details Stored in a SAS Data Set

The SAS data set needs to contain an ID that is equivalent to the name option in DISCRETEATTRMAP. In addition, it needs to have a record for each of the possible values in the data that will have specific attributes. A variable is needed for each attribute that is to be specified. For example, if you want to specify marker symbols and marker colors as well as line pattern and line color, then you would have variables MARKERSYMBOL, MARKERCOLOR, LINEPATTERN and LINECOLOR to capture these attributes. Only the attributes that are needed are required to be defined in the data set. Once the data set is created then it needs to be associated with the template and the data. This is done in during the rendering process where the attribute map data set is specified using DATTRMAP option (SAS Program 9).

```
proc sgrender data = all2 template = ptprof
              dattrmap = colormaps;
run;
```

SAS Program 9. Specifying Attribute Map Data Set on PROC SGRENDER

CONCLUSION

There are many types of plots with various types of options. The types of graphs that can be produced are limitless. We have only touched upon a very small portion of what GTL is capable of. SAS and GTL provide the tools to see how a graph can be broken down into different pieces, and then combined to create the final output, making a task that may have at one time seemed impossible, possible.

ACKNOWLEDGEMENTS

I would like to thank Kriss Harris for his thorough review of the content and providing valuable feedback. I would also like to thank Louise Hadden for acting as my editor and helping clean up grammar and punctuation and ensuring that the explanations were clear to a non-GTL user.

REFERENCES

Graph Template Language Tip Sheet. (2018, 03 13). Retrieved from https://support.sas.com/rnd/app/ODSGraphics/TipSheet_GTL.pdf

Graph Template Modification Tip Sheet. (2018, 03 13). Retrieved from https://support.sas.com/rnd/app/ODSGraphics/TipSheet_GraphTemplateModification.pdf

Harris, Kriss. 2017. "Hands-on Graph Template Language: Part B". *Proceedings of SAS Global Forum 2017.* Orlando, FL: SAS Institute, Inc. http://support.sas.com/resources/papers/proceedings17/0864-2017.pdf

Harris, Kriss and Richann Watson. (2018). "Great Time to Learn GTL". *Proceedings of PharmaSUG*. Seattle, WA: PharmaSUG. https://www.lexjansen.com/pharmasug/2018/EP/PharmaSUG-2018-EP18.pdf

Matange, Sanjay. *Getting Started with the Graph Template Language in SAS®*, SAS Institute (SAS Press, 2013).

SAS® 9.4 Graph Template Language: Reference, Fifth Edition. (2018, 03 13). Retrieved from http://documentation.sas.com/?docsetId=grstatgraph&docsetTarget=p1rdkldsdjjotln1v88o3rdglyb7.htm&docsetVersion=9.4&locale=en

SAS® 9.4 Graph Template Language: User's Guide, Fifth Edition. (2018, 03 13). Retrieved from http://documentation.sas.com/?docsetId=grstatug&docsetTarget=titlepage.htm&docsetVersion=9.4&locale=en

RECOMMENDED READING

Matange, Sanjay. *Getting Started with the Graph Template Language in SAS®*, SAS Institute (SAS Press, 2013).

Matange, Sanjay. *Clinical Graphs Using SAS®*, SAS Institute (SAS Press, 2016).

Kuhfeld, Warren. *Basic ODS Graphics Examples.* Free download, http://support.sas.com/documentation/prod-p/grstat/9.4/en/PDF/odsbasicg.pdf

Kuhfeld, Warren. *Advanced ODS Graphics Examples.* Free download, https://support.sas.com/documentation/prod-p/grstat/9.4/en/PDF/odsadvg.pdf

REFERENCE LINKS

SAS Predefined Colors:
https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.2&docsetId=grstatug&docsetTarget=n161ukdyz9wpfsn1nh8sihforvyq.htm&locale=en

SAS/Graph Font Lists:
https://documentation.sas.com/?docsetId=graphref&docsetTarget=n0c8945h7o2kmrn1h0uehmio2i6j.htm&docsetVersion=9.4&locale=en

SAS Available Line Patterns
https://documentation.sas.com/?cdcId=pgmsascdc&cdcVersion=9.4_3.2&docsetId=grstatug&docsetTarget=n13pm0ndse66l2n1u309543mx2yt.htm&locale=en

SAS Marker Attributes and Symbols
https://documentation.sas.com/?docsetId=grstatproc&docsetTarget=p0i3rles1y5mvsn1hrq3i2271rmi.htm&docsetVersion=9.4&locale=en

CONTACT INFORMATION

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

Richann Watson
DataRich Consulting
richann.watson@datarichconsulting.com
http://www.datarichconsulting.com