MWSUG 2018 - Paper PO-032

Great Time to Learn GTL

Kriss Harris, SAS Specialists Limited; Richann Watson, DataRich Consulting

ABSTRACT

It's a Great Time to Learn GTL! Do you want to be more confident when producing GTL graphs? Do you want to know how to layer your graphs using the OVERLAY layout and build upon your graphs using multiple LAYOUT statement? This paper guides you through the GTL fundamentals!

INTRODUCTION

GTL is part of the ODS Graphics software that is utilized by several SAS procedures. Although these SAS procedures (e.g., the SGPLOT procedure) have pre-defined graph templates which produce outstanding graphs, they may not produce the type of output that is desired. There may be times when several of the features in the pre-defined graph templates need to be modified in order to produce the desired output. GTL will allow the user to modify features for graphs that are based on pre-defined templates, and it will allow users to create graphs that cannot be produced from a pre-defined template. In addition, GTL makes it easier to incorporate features, such as embedding a table of data or displaying different graphs on the same page, that may have been difficult to incorporate previously. Furthermore, 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. However, if those are not sufficient reasons, then according to Matange (2013, p. 5) some additional reasons to learn GTL are:

- GTL provides in one system the full set of features that you need to create graphs from the simplest scatter plots to complex diagnostics panels.
- GTL is the language used to create the templates shipped by SAS for the creation of the automatic 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 release.

The data set used in this paper is from the CDISC SDTM / ADaM Pilot Project and this was obtained from the CDISC website (CDISC, 2013).

THE BASICS

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

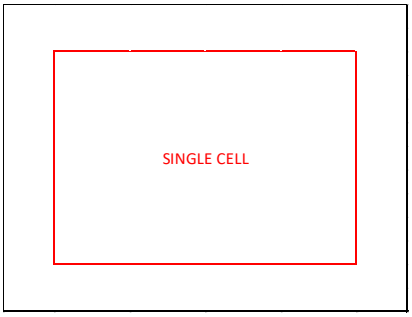
LAYOUTS

Types

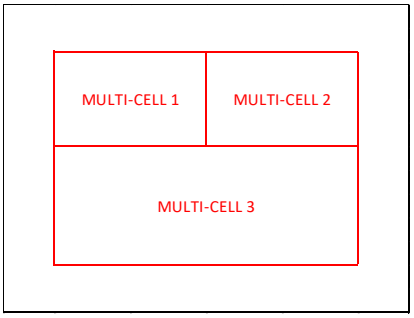
Within GTL there are several layouts. These layouts can be classified into one of 3 categories: single cell, multi-cell pre-defined and multi-cell data-driven. Within each type, there are various other layouts that will help further refine what type of display is needed.

- 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 layouts mean and how they can be utilized to produce a graph is a key part of the use of GTL. The desired output will determine what layout should be used.

General Syntax

Regardless of the type of layout selected, each layout is initiated with LAYOUT and terminated with ENDLAYOUT.

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

```
layout type </options>;
      …
endlayout;
```

The syntax for LATTICE has 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 creates axes that are determined by plots in each cell. However, if you want to have uniform axes the COLUMNDATARANGE and ROWDATARANGE options on the layout statement can be utilized. Further control of axes, can be done by specifying COLUMNAXES and COLUMN2AXES to control the X and X2 axes, respectively and ROWAXES and ROW2AXES 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>;
      …
      <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;
```

PLOTS

There are a variety of plots to choose from. Each plot will have its own syntax (Table 1) and own set of options (Table 2) available to it.

PLOT	SYNTAX
BARCHART	BARCHART CATEGORY = *column* | *expression* *</option(s)>;* BARCHART CATEGORY = *column* | *expression* RESPONSE = *numeric-column* | *expression* *</option(s)>;*
BOXPLOT	BOXPLOT Y = *numeric-column* | *expression* *</option(s)>;* BOXPLOT X = *column* | *expression* Y = *numeric-column* | *expression* *</option(s)>;*
SCATTERPLOT	SCATTERPLOT X = *column* | *expression* Y = *column* | *expression* *</option(s)>;*

Table 1. Syntax for Select Plots

OPTION	ALLOWED VALUES	BARCHART	BOXPLOT	SCATTERPLOT
APPEARANCE				
BARWIDTH	*number*	✓		
BOXWIDTH	*number*		✓	
CAPSHAPE	SERIF| LINE | BRACKET | NONE		✓	
CLUSTERWIDTH	*number*	✓	✓	✓
COLORBYFREQ	TRUE | FALSE	✓		
COLORMODEL	* *color-ramp-style-element* | *(color-list)* ** *style-element* | *(color-list)*	✓*		✓**
COLORRESPONSE	*numeric-column* | *range-attr-var* | *expression*	✓		✓
DATASKIN	NONE | CRISP | GLOSS | MATTE | PRESSED | SHEEN	✓	✓	✓
DATATRANSPARENCY	*number*	✓	✓	✓
DISCRETEMARKERSIZE	*number*			✓
DISPLAY	STANDARD | ALL | *(display-options)*	✓	✓	
EXTREME	TRUE | FALSE		✓	
MARKERATTRS	*style-element* | *style-element (marker-options)* | *(marker-options)*			✓
AXIS				
XAXIS	X | X2	✓	✓	✓
YAXIS	Y | Y2	✓	✓	✓
LABEL				
BARLABEL	TRUE | FALSE	✓		
BARLABELATTRS	*style-element* | *style-element (text-options)* | *(text-options)*	✓		
BARLABELFITPOLICY	AUTO | NONE	✓		
BARLABELFORMAT	*format*	✓		
DATALABEL	* *column* ** *column* | *expression*		✓*	✓**
DATALABELATTRS	*style-element* | *style-element (text-options)* | *(text-options)*		✓	✓
DATALABELPOSITION	AUTO | TOPRIGHT | TOP | TOPLEFT | LEFT | CENTER | RIGHT | BOTTOMLEFT | BOTTOM | BOTTOMRIGHT			✓
DATALABELSPLIT	TRUE | FALSE		✓	✓
DATALABELSPLITCHAR	*"character-list"*		✓	✓
LEGENDLABEL	*"string"*	✓	✓	✓

Table 2. Sample of Options Available for Select Plots

For a complete list of plots and all features available visit Plot Statements → Plot Statements at http://support.sas.com/documentation/cdl/en/grstatgraph/69718/HTML/default/viewer.htm#p1rdkldsdjjotln1v88o3rdglyb7.htm.

HOW DOES GTL COMPARE TO SG PROCEDURES?

If you have used SGPLOT, then you are part way there to understanding the GTL syntax. There are some differences in syntax which are illustrated using a few common types of plots in Table 3. In addition, there are some subtle differences in the more common options used as show in Table 4.

In these examples, one of the main differences is the use of an argument named *expression*. *Expression* allows you to plot data that is not found in the data set. For example, in your plot statement instead of specifying that the variable X = *column_x* (a variable in your data set), you could use an expression to specify that X is column_x + 5, which could be done by using the syntax X = *eval(column_x + 5)*. For details on the utilization of an expression on a plot statement please refer to Harris, 2017, p. 8, 13.

PLOTS	GTL SYNTAX	SGPLOT SYNTAX
Scatter Plot	SCATTERPLOT X = *column* | *expression* Y = *column* | *expression </option(s)>;*	SCATTER X=*variable* Y = *variable* </option(s)>
Series Plot	SERIESPLOT X = *column* | *expression* Y = *column* | *expression </option(s)>;*	SERIES X = variable Y = variable </option(s)>

Table 3. Examples of Comparisons Between Plot Statements in SGPLOT and GTL with Similarities

OPTIONS	GTL SYNTAX	SGPLOT SYNTAX
Change x-axis label	XAXISOPTS = (label = "New Label")	XAXIS label = "New Label";
Change y-axis range	YAXISOPTS = (linearopts = (viewmin = 0 viewmax = 100))	YAXIS min = 0 max = 0;
Specify tick values	YAXISOPTS = (linearopts = (tickvaluesequence = (start=0 end=0 increment=10)))	YAXIS values = (0 to 100 by 10);

Table 4. Examples of Comparisons Between Options in SGPLOT and GTL

Although these are simple examples that illustrate the similarities between SGPLOT and GTL not all plots will have the same ease of transference of syntax as demonstrated in Table 5.

With some plots there will be more options to choose from and these options could be based on orientation (i.e., portrait or landscape) or what type of layout is used (e.g., DATAPANEL or DATALATTICE).

PLOT	GTL SYNTAX	SGPLOT SYNTAX
BARCHART	BARCHART CATEGORY = *column* | *expression </option(s)>;* BARCHART CATEGORY = *column* | *expression* RESPONSE = *numeric-column* | *expression* *</option(s)>;* BARCHARTPARM CATEGORY = *column* | *expression* RESPONSE=*numeric-column* | *expression </option(s)>*;	VBAR *category-variable* </option(s)>
BOXPLOT	BOXPLOT Y = *numeric-column* | *expression </option(s)>;* BOXPLOT X = *column* | *expression* Y = *numeric-column* | *expression </option(s)>;* BOXPLOTPARM Y = *numeric-column* | *expression* STAT = *string-column </option(s)>;* BOXPLOTPARM X = *column* | *expression* Y = *numeric-column* | *expression* STAT = *string-column </option(s)>;*	VBOX *analysis-variable* </option(s)>

Table 5. Examples of Comparisons Between Plot Statements in SGPLOT and GTL without Similarities

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. Second you need to associate the data 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 SAS code …

      endlayout;
   endgraph;
end;
run;
```

(1) Define the structure of the graph by creating a custom template. Defining a custom template is done using STATGRAPH and providing a template name (***templatename***). The template name is used when rendering the graph. This statement has a corresponding END.

(2) Each custom template definition has at most 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. For each LAYOUT specified a corresponding ENDLAYOUT needs to be used to signal the end of that particular layout. Note that it is possible to nest layouts depending on the type of layout.

(4) Within each LAYOUT block, the required plot statement(s) and options are specified. Depending on the type of layout(s) and plot(s) will drive how this portion is programmed.

2. PRODUCE THE GRAPH

As Matange indicated, 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 as long as 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

Using data from CDISC SDTM / ADaM Pilot Project, we illustrate how easy it is to produce a graph. In this example, we illustrate two ways to produce a barchart by treatment with the statistics displayed above each bar. In addition, we want the graph to contain a table with comparison statistics and a trend test. Figure 1 illustrates the graph that needs to be produced.

SINGLE CELL LAYOUT WITH NESTED LAYOUT

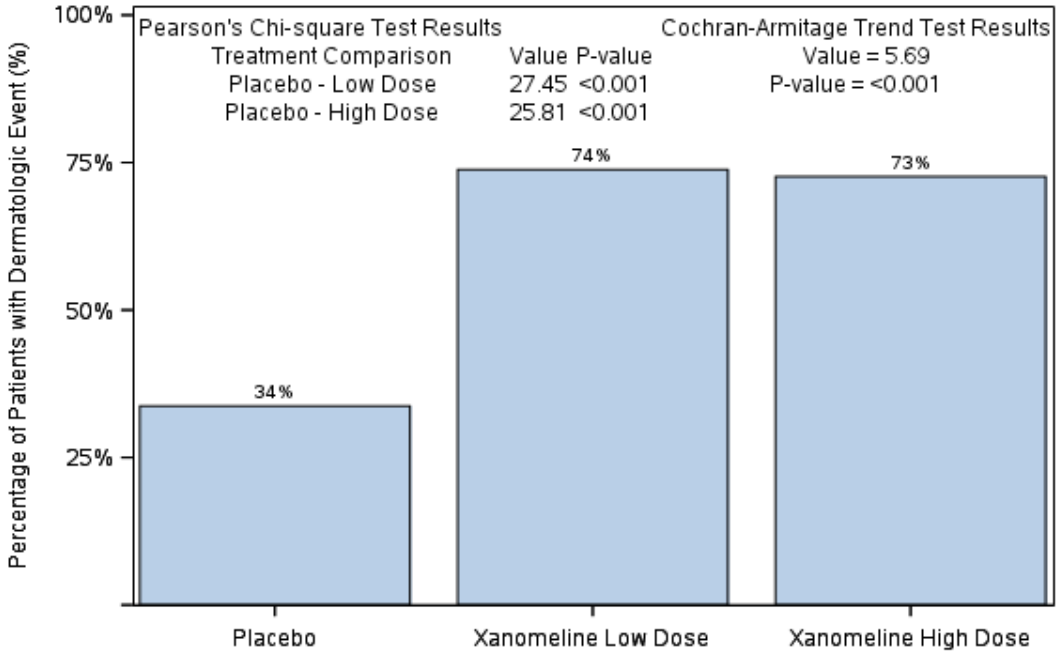


Figure 1. Bar Chart with Embedded Table Using GRIDDED Layout Nested in OVERLAY Layout

In this specific layout it is evident that there is only one cell in the graph area. Although it may at first seem like there is more than one portion of the graph, in reality there is only one portion. This is evident by looking at the y-axis. The embedded table is contained within the 75-100% portion of the barchart. This approach uses a single cell layout (OVERLAY) but has a nested layout (GRIDDED) within it. The outer layout will contain the necessary statements to produce the main portion of the graph (i.e., the bar chart) and the inner layout will contain the necessary statements to embed the table within the bar chart graph area. Refer to SAS Program 1 for complete code and explanation of the various options.

```
proc template;
   define statgraph recrgrphg;

     /* indicate the macro variables that will be used to create the inset table */
     mvar valuechi054 pchi054 valuechi081 pchi081 cmstat cmpvalue;

      begingraph / border = false;

        /* need to force the y-axis to display up through */
        /* 100 in order for the table to be displayed     */
        layout overlay / xaxisopts = (label = " "
                                      type = discrete)
                         yaxisopts = (label = "Percentage of Patients with
                                              Dermatologic Event (%)"
                                      linearopts = (tickvaluesequence =
                                                    (start=0 end=100 increment=25)
                                                     viewmin=0 viewmax=100));

          /* create the vertical bar charts for each treatment group */
          barchart x = TRTAN y = PCT_ROW / orient = vertical barlabel = true;

          /* portion to embed table - to be placed at the top            */
          /* order = rowmajor indicates that the grid will be filled out */
          /* in row order so each column in the row will be populated    */
          /* prior to moving onto the next row                           */
          layout gridded / columns = 4
                           order = rowmajor
                           autoalign = (top);
                           entry "Pearson's Chi-square Test Results";
                           entry " ";
                           entry " ";
                           entry "Cochran-Armitage Trend Test Results";
                           entry "    Treatment Comparison";
                           entry "Value";
                           entry "P-value";
                           entry "    Value = " cmstat;
                           entry "    Placebo - Low Dose";
                           entry valuechi054;
                           entry pchi054;
                           entry "P-value = " cmpvalue;
                           entry "    Placebo - High Dose";
                           entry valuechi081;
                           entry pchi081;
          endlayout;
        endlayout;
      endgraph;
   end;
 run;

proc sgrender data = PCT template = recrgrphg;
  format TRTAN trt. PCT_ROW pctfmt.;
run;
```

SAS Program 1. PROC TEMPLATE to Produce Bar Chart with an Embedded Table Using GRIDDED Layout Nested in OVERLAY Layout

1 Since the values of the statistics in the embedded table can vary based on the data source, it may be beneficial to store the values within a macro variable. When using macro variables within the template then the MVAR or MVARN statement should be used. Macro variables that are declared using MVAR will resolve to a string where macro variables declared with MVARN will convert to a numeric token. The use of MVAR(N) allows the macro variable to be resolved at execution rather than a compile time like it would

be if the ampersand (&) is used. This allows the template to be defined prior the macro variables being defined.

(2) Within each type of layout there are various options. For this example, we illustrate some options associated with the axes. For the x-axis, we indicate the data is discrete and that we do not want a label associated with the variable to be displayed. For the y-axis, we want to display a specific label instead of the label associated with the variable. In addition, we want to force the y-axis to go from 0 to 100 in increments of 25 regardless of what the data produces. Furthermore, the VIEWMIN and VIEWMAX options are used to indicate that only data within that range should be displayed. Since this example is for percentages, these options could be eliminated. However, these options are beneficial when there is a possibility for data to have outliers and you want to eliminate the outliers from showing on the graph. This does not change the data it just changes what was displayed.

(3) Within the OVERLAY layout, the necessary plot statement along with the necessary options is specified to build the bar chart. For this example, we want the bars to be vertical and we want each bar to be labelled with the value.

(4) Nesting the GRIDDED layout within the OVERLAY allows the table to be embedded directly into the graph area. Defining the number of columns and specifying the order in which the grid will be filled out will produce the desired table within the graph. Using our example, the grid will be filled out in row order so that each of the four columns is filled out prior to moving to the next row. Each portion of the table is then entered as an entry. If a specific cell in the table should be left null, then a blank entry line should be created so that when the template is rendered it will not populate that particular cell in the table. For example, `entry "  Value = " cmstat;` is displayed on the second row, fourth column; while `entry valuechi054;` and `entry pchi054;` are on the third row, third entry and fourth entry respectively.

(5) When the graph is rendered, the order of the treatment groups is important so rather than using the character version of the treatment values, the treatment groups are assigned a numeric counterpart and then a format is applied to the numeric value to display the actual treatment. In addition, a format was applied to the percent value so that it would display with the percent sign. **Error! Reference source not found.**Figure 1 shows the rendering of the code in SAS Program 1.

MULTI-CELL LAYOUT USING LATTICE AND INDIVIDUAL LAYOUTS WITHIN EACH CELL

What if some of the percentages in the bar chart are more than 75% and there is no space within the single-cell layout to embed the table, then an alternative approach needs to be considered. By splitting the graph area into multiple cells, will allow the bar chart to be displayed up through 100% if necessary as well as display the necessary statistics.

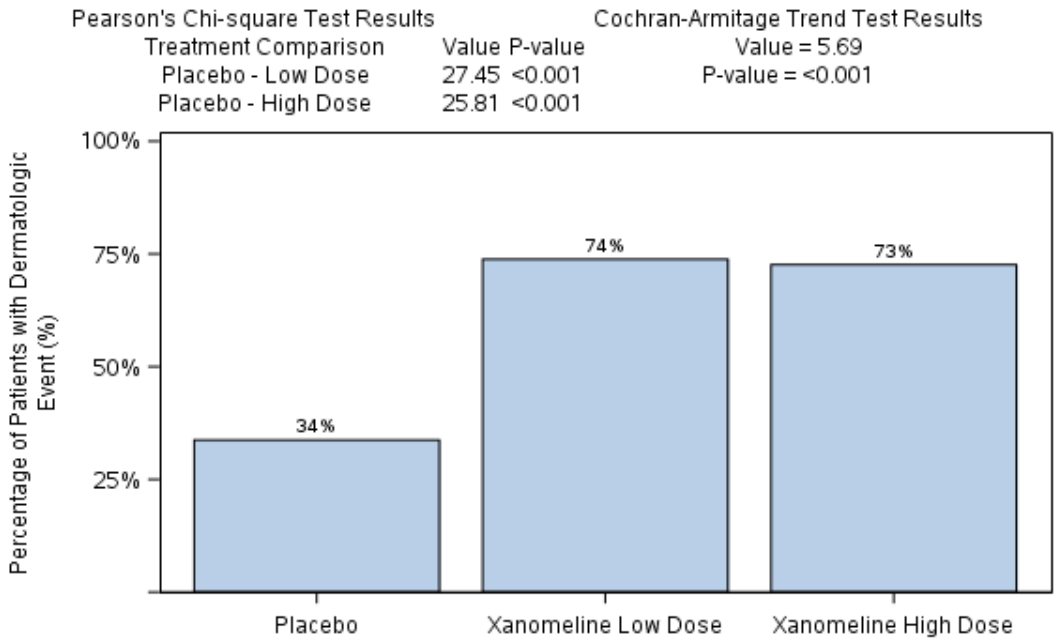


Figure 2. Bar Chart by Treatment with Embedded Table Using LATTICE Layout and Individual Layouts Within Each Cell

```
proc template;
   define statgraph recrgrph;

      /* indicate the macro variables that will be used to create the inset table */
      mvar valuechi054 pchi054 valuechi081 pchi081 cmstat cmpvalue;
      begingraph / border = false;
           layout lattice / rows = 2 rowweights = (0.2 0.8);  (1)

                layout overlay / yaxisopts=(display=none)  yaxisopts=(display=none);
                   /* portion to embed table - 4 columns entered in row order */
                   layout gridded / columns = 4
                                order = rowmajor
                                autoalign = (top);
                                entry "Pearson's Chi-square Test Results";
                                entry " ";
                                entry " ";
                                entry "Cochran-Armitage Trend Test Results";
                                entry "    Treatment Comparison";
                                entry "Value";
                                entry "P-value";
                                entry "    Value = " cmstat;
                                entry "    Placebo - Low Dose";
                                entry valuechi054;
                                entry pchi054;
                                entry "P-value = " cmpvalue;
                                entry "    Placebo - High Dose";
                                entry valuechi081;
                                entry pchi081;
                   endlayout;
                endlayout;
```

(2)

```
                layout overlay / xaxisopts = (label = " " type = discrete)
                                 yaxisopts = (labelsplitchar="#"
                                              labelfitpolicy=splitalways
                                              label = "Percentage of Patients with
Dermatologic#Event (%)"
                                              linearopts = (tickvaluesequence =
                                                            (start = 0 end = 100
                                                                    increment = 25)
                                                             viewmin = 0 viewmax =100));

                    /* create the vertical bar charts for each treatment group */
                    barchart x = TRTAN y = PCT_ROW / orient = vertical barlabel = true;
                endlayout;
            endlayout;
        endgraph;
    end;
run;
```

SAS Program 2. PROC TEMPLATE to Produce a Bar Chart by Treatment with an Embedded Table Using LATTICE Layout and Individual Layouts Within Each Cell

1. LATTICE allows you to split area into rows or columns or a combination of rows and columns based on the desired output. For our example, we split graph area into two portions by rows with the first row being 20% of area and the second row is 80%.

2. With the table being the top portion of the graph, layout for the table is specified first and will be displayed in the 20% of the graph area. The syntax is the same as it is in the single-cell layout example. The difference is that the GRIDDED layout is nested within the LATTICE layout and not the OVERLAY layout.

3. The bar chart will be displayed in the 80% of the graph area. The syntax is the same as it is in the single-cell layout example with the nested GRIDDED layout removed.

4. Since the area specified for the bar chart is smaller than the one in the single-cell layout the label may not fit, thus we introduce the use of the LABELFITPOLICY option. This will indicate how the label should be displayed if it does not fit in the allotted space. The default option is AUTO which indicates that if a short label is provided and will fit within the allotted space then the short label will be used; otherwise the short label if provided is clipped and if the short label is not provided the original label is clipped. If LABELFITPOLICY = SPLITALWAYS then the label will **always** be split when it encounters a split character. Therefore, LABELSPLITCHAR needs to be specified to indicate what the split character is. Note that the default for LABELSPLITCHAR is a blank space, therefore, when LABELFITPOLICY = SPLITALWAYS it is necessary to specify the non-blank split character. For our example, we used LABELFITPOLICY = SPLITALWAYS which indicates that the label will be split when it encounters the specified split character. Note that the LABELFITPOLICY is only effective when LABELPOSITION is CENTER or DATACENTER. The default for LABELPOSITION is CENTER.

CONCLUSION

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

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, K. (2017). Hands-on Graph Template Language: Part B . *SAS Global Forum 2017.* Orlando: SAS Global Forum.

Available at http://support.sas.com/resources/papers/proceedings17/0864-2017.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 READINGS

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

CONTACT INFORMATION

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

Kriss Harris
SAS Specialists Limited
italjet125@yahoo.com
http://www.krissharris.co.uk

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