

Hands-On Graph Template Language (GTL): Part A

Kriss Harris, SAS Specialists Limited, Hertfordshire, United Kingdom

ABSTRACT

Would you like to be more confident in producing graphs and figures? Do you understand the differences between the OVERLAY, GRIDDED, LATTICE, DATAPANEL, and DATALATTICE layouts? Finally, would you like to learn the fundamental Graph Template Language methods in a relaxed environment that fosters questions? Great—this topic is for you! In this hands-on workshop, you are guided through the fundamental aspects of the GTL procedure, and you can try fun and challenging SAS® graphics exercises to enable you to more easily retain what you have learned.

INTRODUCTION

Creating sophisticated graphs for your deliverables is one reason for you to know how to use Graph Template Language (GTL). Another reason is because there are some graphs which only GTL can create. According to (Matange, Getting Started with the Graph Template Language in SAS: Examples, Tips, and Techniques for Creating Custom Graphs, 2013, p. 5) some 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.

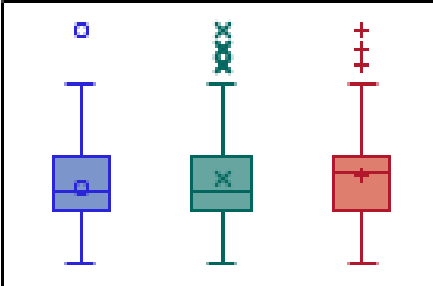
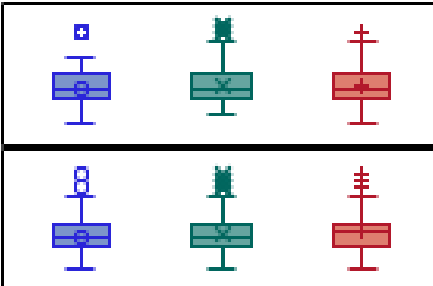
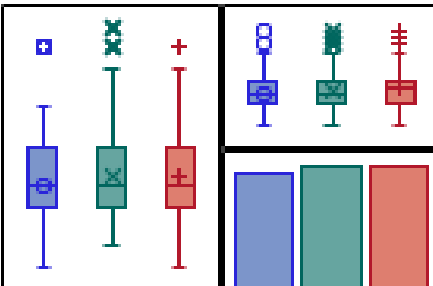
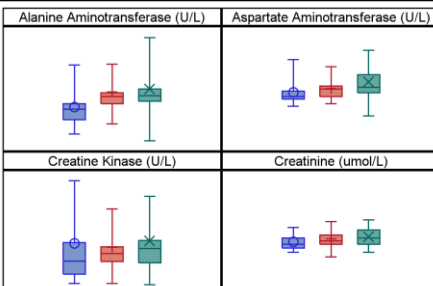
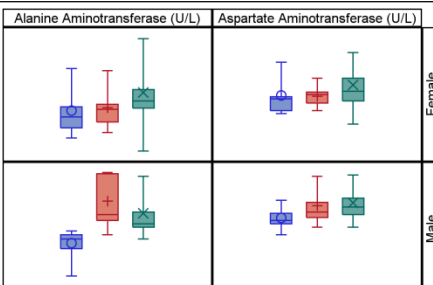
This paper intends to motivate you to use GTL. You will be introduced to GTL, and shown the best layouts to use and why. SAS® 9.4 was used to run the code in this paper.

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

LAYOUTS

The OVERLAY, LATTICE, DATAPANEL and PROTOTYPE layouts can be used to produce the majority of your plots, if not all of your plots. Table 1 below, summarizes those layouts, and includes the GRIDDED and DATALATTICE layout.

Table 1: Various Layouts in GTL

Layouts	Example Plot
<p>OVERLAY layout – This layout is equivalent to using SGPLOT. It's for plotting your single-cell plots. Such as plotting a boxplot or a scatterplot, etc.</p>	
<p>GRIDDED layout – Use this layout to produce multi-cell plots which have the same proportion in width or height. For example, in the figure on the right, the rows are the same size.</p> <p>Summary statistics can also be plotted with the GRIDDED layout.</p> <p>The GRIDDED layout needs to be used in conjunction with the OVERLAY layout.</p>	
<p>LATTICE layout – This layout is very flexible and can be used to produce multi-cell plots which have different heights and widths. It's possible to also nest layouts within the LATTICE layout.</p> <p>The LATTICE layout needs to be used in conjunction with the OVERLAY layout.</p>	
<p>DATAPANEL layout – This layout is equivalent to using SGPANEL. Use this layout to produce your multi-cell plots, which are paneled by your variable(s) of choice. In the example on the right, the plots are paneled by each Laboratory test.</p> <p>The DATAPANEL layout needs to be used in conjunction with the PROTOTYPE layout.</p>	
<p>DATALATTICE layout – This layout is equivalent to using SGPANEL with the LAYOUT=LATTICE option. Use this layout to produce your multi-cell plots, when you have exactly two classification variables. In the example on the right, the plots are paneled for each Laboratory test and for each Gender.</p> <p>The DATALATTICE layout needs to be used in conjunction with the PROTOTYPE layout.</p>	
<p>PROTOTYPE layout – This layout is essentially a restricted OVERLAY layout with the same general rules for overlaying plots. This is used with DATAPANEL and DATALATTICE.</p>	

FROM SG PROCEDURES TO GTL

If you are familiar with SG PLOT then you will probably find it easier to understand the GTL syntax. Literally it is like learning a new language! Table 2 below shows the relationships in the syntax between SG PLOT and GTL in the plot statements and Table 3 shows options that you are most likely to use. The general background color of Table 2 represents the plot type that the plot would be in SG PLOT. In this context the general background color means that the red and pink color are both referred to as red, and similar groupings are done with the other colors. The red background represents the **basic plots** type, blue represents **fit and confidence plots**, purple represents the **categorization plots**, and olive green represents the **distribution plots**.

You will notice that the **basic plots** and **fit and confidence plots** work quite similar in GTL as they did in SG PLOT. In most of the plot types all that is needed is to add "PLOT" on the end. You will also notice that in the GTL syntax there is also an argument called expression. An expression allows you to plot data that is not in actually in the dataset, i.e. it derives the data on the fly. For example, in your plot statement instead of specifying that the variable $X = \text{column_x}$ (a variable in your dataset), you could use an expression to specify that X is $\text{column_x} + 5$, which could be done by using the syntax $X = \text{eval}(\text{column_x} + 5)$. (Harris, 2016, p. 16) shows an example of how to use expressions.

In GTL the **categorization plots** and the **distribution plots** are slightly more difficult when compared to the **basic plots** and **fit and confidence plots**. You will notice that there are generally more options such as the BARCHART and BARCHARTPARM, HISTOGRAM and HISTOGRAMPLOT statements. For **categorization plots** and **distribution plots**, the plot statement to use depends on the layout that is used. Generally if you use the PROTOTYPE layout, which you will have to use if you use the DATAPANEL or DATALATTICE layout, then the equivalent statement with "PARM" at the end should be used. This is because the PROTOTYPE layout expects summarized data. More details on the PROTOTYPE layout are found later on in the paper.

Table 2: Examples of Comparisons between plot statements in SG PLOT and GTL

Plots	SGPLOT syntax	GTL Syntax
Scatter plot	SCATTER X= <i>variable</i> Y= <i>variable</i> </option(s)>	SCATTERPLOT X= <i>column</i> <i>expression</i> Y= <i>column</i> <i>expression</i> </option(s)>
Series plot	SERIES X= <i>variable</i> Y= <i>variable</i> </option(s)>	SERIESPLOT X= <i>column</i> <i>expression</i> Y= <i>column</i> <i>expression</i> </option(s)>
Regression Plot	REG X= <i>numeric-variable</i> Y= <i>numeric-variable</i> </option(s)>	REGRESSIONPLOT X= <i>numeric-column</i> <i>expression</i> Y= <i>numeric-column</i> <i>expression</i> </<regression-options> <option(s)>>
Loess plot	LOESS X= <i>numeric-variable</i> Y= <i>numeric-variable</i> </option(s)>	LOESSPLOT X= <i>numeric-column</i> <i>expression</i> Y= <i>numeric-column</i> <i>expression</i> </<regression-options> <option(s)>>
Bar Chart	VBAR <i>category-variable</i> </option(s)>	BARCHART CATEGORY= <i>column</i> <i>expression</i> </option(s)> BARCHART CATEGORY= <i>column</i> <i>expression</i> RESPONSE= <i>numeric-column</i> <i>expression</i> </option(s)> BARCHARTPARM CATEGORY= <i>column</i> <i>expression</i> RESPONSE= <i>numeric-column</i> <i>expression</i> </option(s)>
Dot plot	DOT <i>category-variable</i> </option(s)>	SCATTERPLOT X= <i>column</i> <i>expression</i> Y= <i>column</i> <i>expression</i> </option(s)>

Plots	SGPLOT syntax	GTL Syntax
Histogram	HISTOGRAM <i>response-variable</i> </option(s)>	HISTOGRAM <i>numeric-column</i> <i>expression</i> </option(s)> HISTOGRAMPARM X= <i>numeric-column</i> <i>expression</i> Y= <i>non-negative-numeric-column</i> <i>expression</i> </option(s)>;
Boxplots	VBOX <i>analysis-variable</i> </option(s)>	BOXPLOT Y= <i>numeric-column</i> <i>expression</i> </option(s)> BOXPLOT X= <i>column</i> <i>expression</i> Y= <i>numeric-column</i> <i>expression</i> </option(s)> BOXPLOTPARM Y= <i>numeric-column</i> <i>expression</i> STAT= <i>string-column</i> </option(s)> BOXPLOTPARM X= <i>column</i> <i>expression</i> Y= <i>numeric-column</i> <i>expression</i> STAT= <i>string-column</i> </option(s)>

Table 3: Examples of Comparisons between option in SGPLOT and GTL

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

CREATING A GRAPH USING GTL

As explained in (McConville & Much, 2015), in GTL, graphs are built by using plot and layout statements. The plot statements determine how data are represented in the graph, and the layout statements determine where the plots are drawn on the graph. More advanced layouts can be used to divide the plot area into multiple independent cells. In addition, statements can be nested so multiple plots can be arranged to create elegant visuals.

As mentioned in (Matange, Getting Started with the Graph Template Language in SAS: Examples, Tips, and Techniques for Creating Custom Graphs, 2013), creating a graph using GTL is a two-step process, which uses both the TEMPLATE and SGRENDER procedures.

Firstly, you define the structure of the graph in the form of the STATGRAPH template using GTL. The typical syntax is shown below.

```
proc template;
  define statgraph <template-name>;
    beginngraph / <options>;
      <GTL statements>;
    endngraph;
  end;
run;
```

Secondly, you associate the data with the template using the SGRENDER procedure to create the graph.

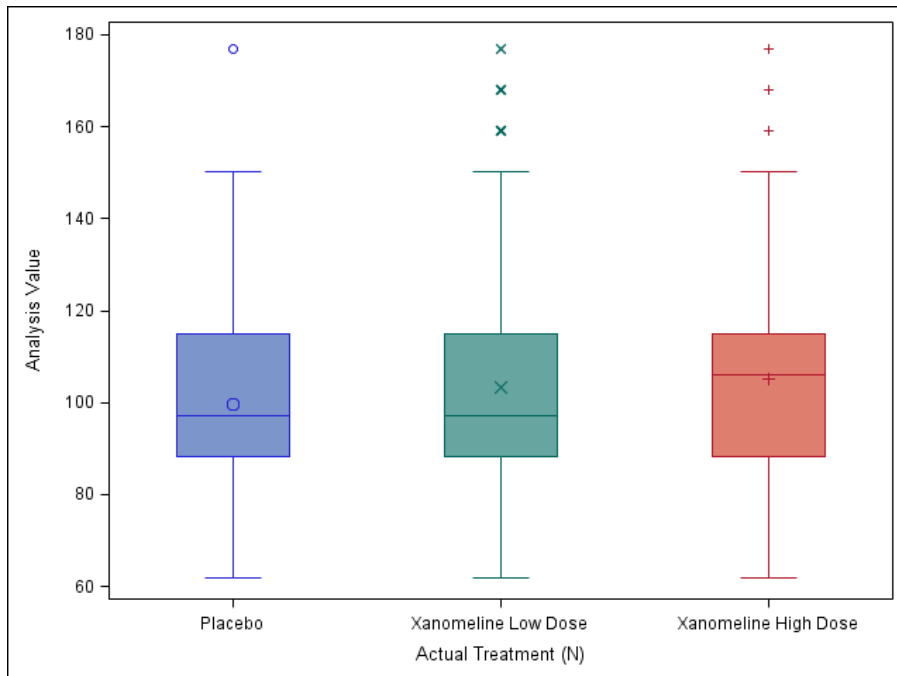
```
proc sgrender data=<data-set-name> template=<template-name>;
  <other optional statements>;
run;
```

The code below, combines the steps above to produce a boxplot of intensity by treatment as seen in Figure 1. You will notice that the code below uses the OVERLAY layout and this is the most commonly used layout for producing single-cell graphs. You can use this layout to maintain the contents of the single-cell, and all plots within this layout share the same area which is bounded by the values used in the x-axis and y-axis. You can use the OVERLAY layout to overlay a number of plot statements on top of one another.

```
proc template;
  define statgraph boxplot_template;
    begingraph;
      layout overlay;
        boxplot x = trtan y = aval / group = trtan groupdisplay = cluster;
      endlayout;
    endgraph;
  end;
run;
```

```
proc sgrender data = adlbc_all template = boxplot_template;
  by param;
  format trtan trtfmt.;
run;
```

Figure 1: Boxplot of Intensity by Treatment Using GTL



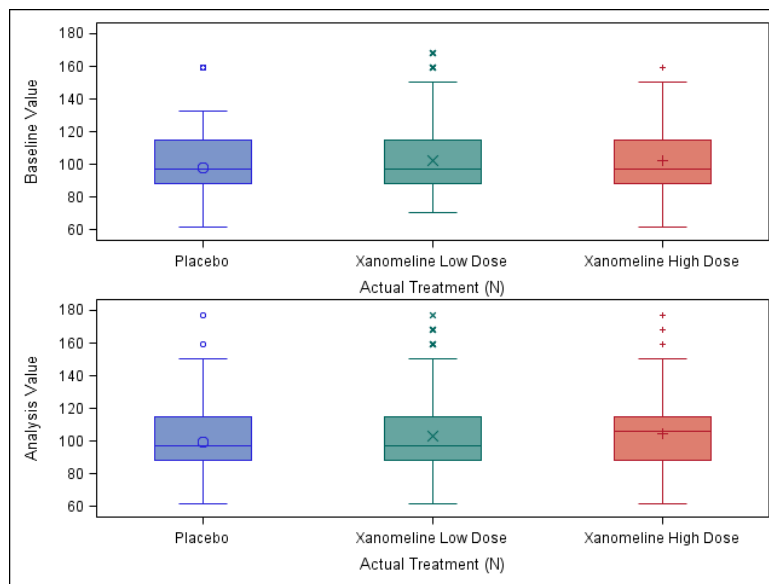
MULTIPLE-CELL GRAPHS

It is quite common to want to see more than one graph on one page of a document or on an image. For example you may want a graph that shows a boxplot and a bar chart on the same figure or you may want to see the distribution of different treatments paneled by a parameter. Examples of this can be seen in Figure 2 and Figure 3, and subsequent figures. Multiple-cell graphs can be achieved by using different layouts. These layouts are the GRIDDED, LATTICE, DATAPANEL and DATALATTICE layout. To use these layouts two other layout types are also needed. One of them you are already familiar with; the OVERLAY layout. The other layout is the PROTOTYPE layout. The GRIDDED layout is the simplest multiple-cell layout to use. All you have to do is wrap the GRIDDED layout around the OVERLAY layout. Figure 2 shows you the output of the default GRIDDED layout, i.e. when no options are used to control the grid. You will notice that equal space has been allocated to the box plots, and that the two boxplots are arranged in one column with two rows. Options were used within the OVERLAY layout to ensure that the y-axis had the same range of values for both plots. For code that was too long to display within the main text of the paper or was deemed unnecessary to share then, `<OPTIONS#>` was used. The exact code for each `<OPTIONS#>` is included in the appendix.

GRIDDED LAYOUT

```
proc template;
  define statgraph boxplot_template;
    begingraph;
      layout gridded;
      * First Row;
      layout overlay / yaxisopts=<OPTIONS1>;
      boxplot x = trtan y = base / group = trtan groupdisplay = cluster;
      endlayout;
      * Second Row;
      layout overlay / yaxisopts=<OPTIONS1>;
      boxplot x = trtan y = aval / group = trtan groupdisplay = cluster;
      endlayout;
    endgraph;
  end;
run;
```

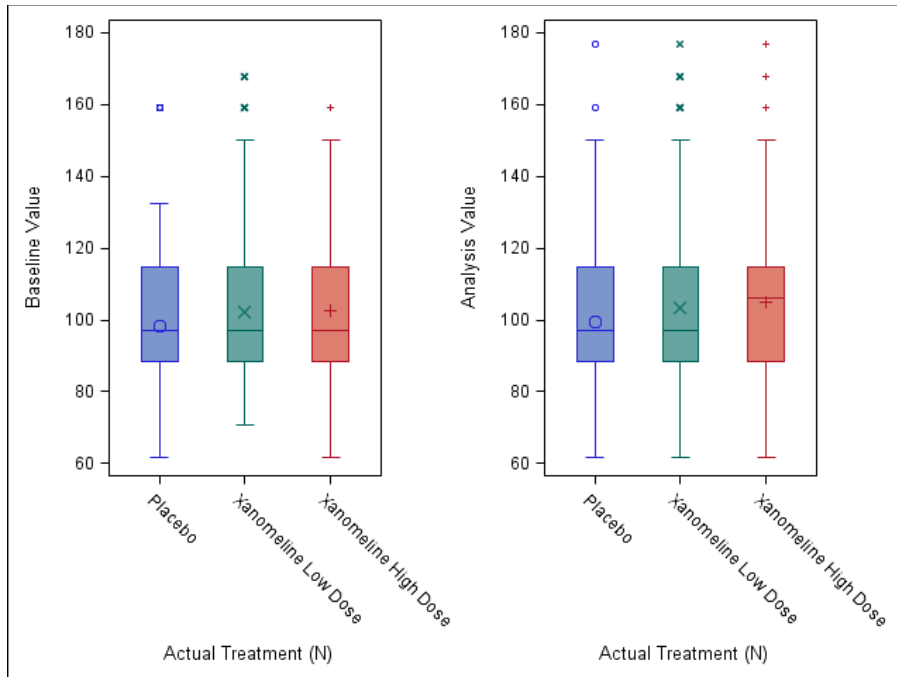
Figure 2: Default Gridded Layout



To arrange the boxplots differently, such as two rows with one column, instead of two columns with one row as in Figure 3, then all that is needed is to add the *columns* option within the GRIDDED LAYOUT.

```
layout gridded / columns = 2;
```

Figure 3: Gridded Layout with Column Option.



LATTICE LAYOUT

The LATTICE layout is more flexible than the GRIDDED layout when it comes to ordering the cells in multiple-cell graphs. For example, with the LATTICE layout you can specify the widths of the heights of the cells, using the *columnweights* and *columnheights* options. The cells do not have to be equal as with the GRIDDED layout. For example, Figure 4 shows the result when the *columnweights* options is used to give different weights to the cells. You will notice that the width of the graph on the left is bigger than the graph on the right.

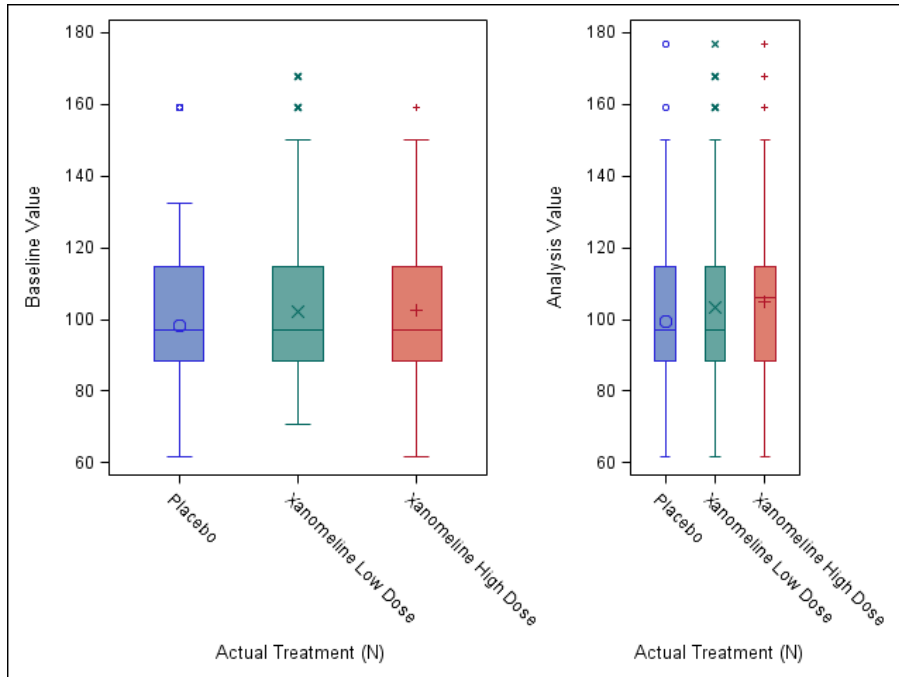
```
proc template;
  define statgraph boxplot_template;
    begingraph;
      layout lattice / columns = 2 columnweights=(0.6 0.4);

      * First Column;
      layout overlay / yaxisopts=(<OPTIONS1>);
      boxplot x = trtan y = base / group = trtan groupdisplay = cluster;
      endlayout;

      * Second Column;
      layout overlay / yaxisopts=(<OPTIONS1>);
      boxplot x = trtan y = aval / group = trtan groupdisplay = cluster;
      endlayout;

    endlayout;
  endgraph;
end;
run;
```

Figure 4: LATTICE Layout with Columnweights Option



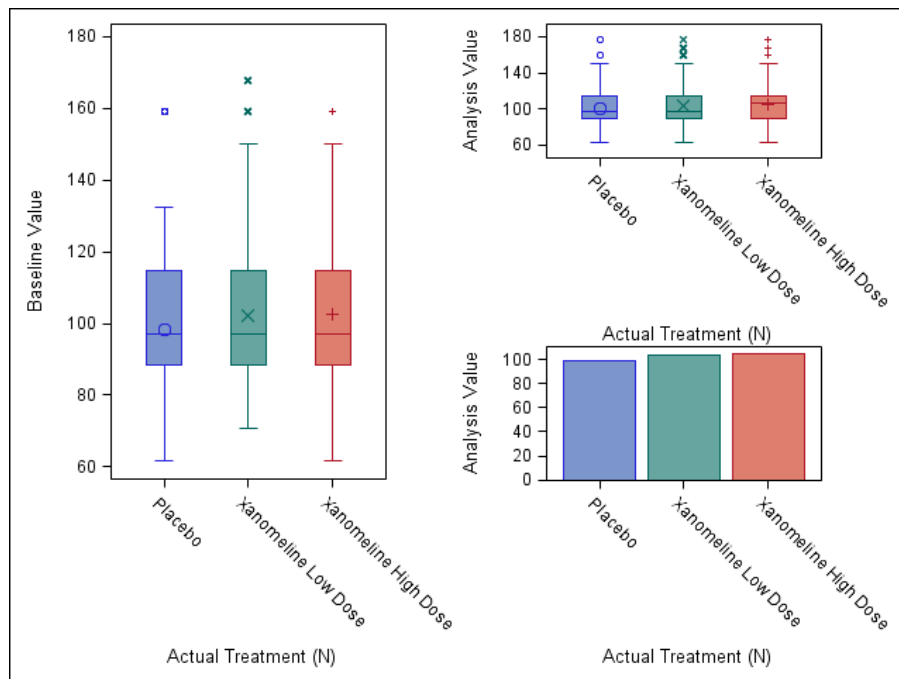
The LATTICE layout can also be used to produce a nested layout as in Figure 5 below. The code below uses two LATTICE layout statements, the first specifies that the graph should have 2 columns, and then the second LATTICE layout statement, specifies that the second column should have 2 rows.

```

begingraph;
  layout lattice / columns = 2 columnweights=(0.5 0.5);
  * First Column;
  layout overlay / yaxisopts=(

```


Figure 5: Nested LATTICE layout



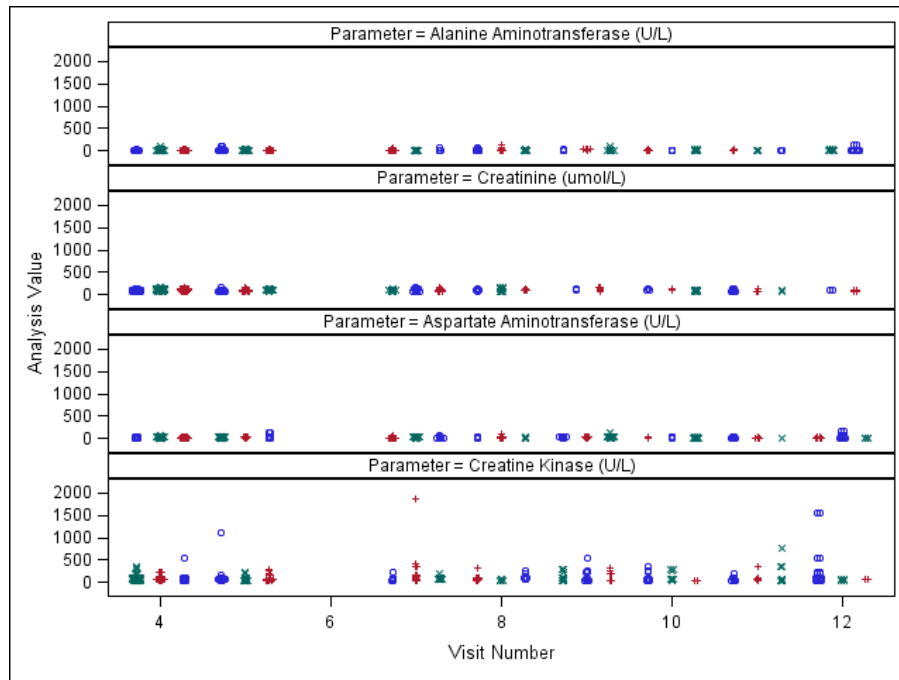
DATAPANEL LAYOUT

The DATAPANEL layout is similar to SGPPANEL. The variable in the **classvars** option determines the number of cells (panels) that the figure has. When using the DATAPANEL layout, you need to use the PROTOTYPE layout with it. The PROTOTYPE layout is essentially a restricted OVERLAY layout with the same general rules for overlaying plots. The main difference between the PROTOTYPE and OVERLAY layout are that there are no axis options available on the LAYOUT PROTOTYPE statement. Axis properties are set with the ROWAXISOPTS= and COLUMNAXISOPTS= options of the parent DATAPANEL or DATALATTICE statement. Another restriction in the PROTOTYPE layout is that only non-computed 2-D plot statements can be used in the LAYOUT PROTOTYPE block. From the plots introduced in Table 2 this means that only BARCHARTPARM, BOXPLOTPARM, HISTOGRAMPARM, SCATTERPLOT and SERIESPLOT can be used in the LAYOUT PROTOTYPE block. This also means that REGRESSIONPLOT, LOESSPLOT, BARCHART, HISTOGRAM and BOXPLOT cannot be used in the LAYOUT PROTOTYPE block because they are computed plot statements, i.e. the data that is displayed on the plots is derived on the fly. An example of the consequences of having to use BOXPLOTPARM instead of BOXPLOT is shown on the next page.

Essentially the LAYOUT PROTOTYPE defines a plot prototype or “rubber stamp” that repeats automatically. Figure 6 below shows the result of using the DATAPANEL and PROTOTYPE layout to produce a scatterplot of intensity by visit number paneled by the laboratory parameter. The *jitter=auto* option allows the y-values to be seen across the x-axis which would have otherwise been obscured due to the similar y-values being overlaid on top of each other.

```
begingraph;
  layout datapanel classvars = (param);
  layout prototype; * Datapanel uses layout prototype;
  scatterplot x = visitnum y = aval / group = trta groupdisplay = cluster
    jitter = auto;
  endlayout;
endlayout;
endgraph;
```

Figure 6: Scatterplot with DATAPANEL



In Figure 6, the y-axis ranges from 0 to 2000 by default, which is due to the range of intensity values in the Creatine Kinase (U/L) laboratory parameter. The maximum range is used by default. Using the maximum range has its benefits and drawbacks. If you want each axis to be consistent for all of the laboratory parameters then using the maximum range is fine, but if the laboratory parameter y-axis values are different, and hence you want each laboratory test to have its own y-axis range then you can use the *rowdatarange=union* option to help you to achieve this. Figure 7 shows the *rowdatarange=union* option in use.

From now on when discussing the DATAPANEL layout, the y-axis will be referred to as row-axis, and the x-axis will be referred to as column-axis, as this is more correct. You will notice that the range of the row-axis makes it difficult to see all the results for each laboratory parameter. Also it appears that showing the intensity values on the log scale and using a different plot such as the boxplot instead of the scatterplot might be more appropriate.

In the *rowaxisopts* you can set the intensity values to be on the log scale, and in the PROTOTYPE layout you can use *boxplotparm* to create boxplots. As mentioned above, the PROTOTYPE layout does not accept computed plot statements. Therefore you cannot use the *boxplot* statement; however, you can use the *boxplotparm* statement. Firstly the dataset needs to be manipulated so that it contains the minimum, maximum, mean, median, 1st quartile and 3rd quartile of the data that you want to plot, which in this case is by treatment and visit number. Then secondly, you plot the data that has already been summarized. In SAS help, there is a generalized macro called *%boxcompute* which can help you to summarize the data.

Figure 7 shows the result of using the *boxplotparm* and shows a boxplot of the intensity by the visit number for each treatment, paneled by the laboratory parameter. The treatment legend is produced by first assigning a name to the *group=trta* values. This is done using the option *name="trtle"*, which then coordinates the colors and the *group* values between the plot and the legend. The DISCRETELEGEND statement within the SIDEBAR block is used to finally plot the legend by using the name which was specified in the *boxplotparm* statement: "trtle".

```

begingraph;
  layout datapanel classvars = (param) / columns = 2
    rowaxisopts = (type = log label = "Lab Intensity Values") rowdatarange=union;

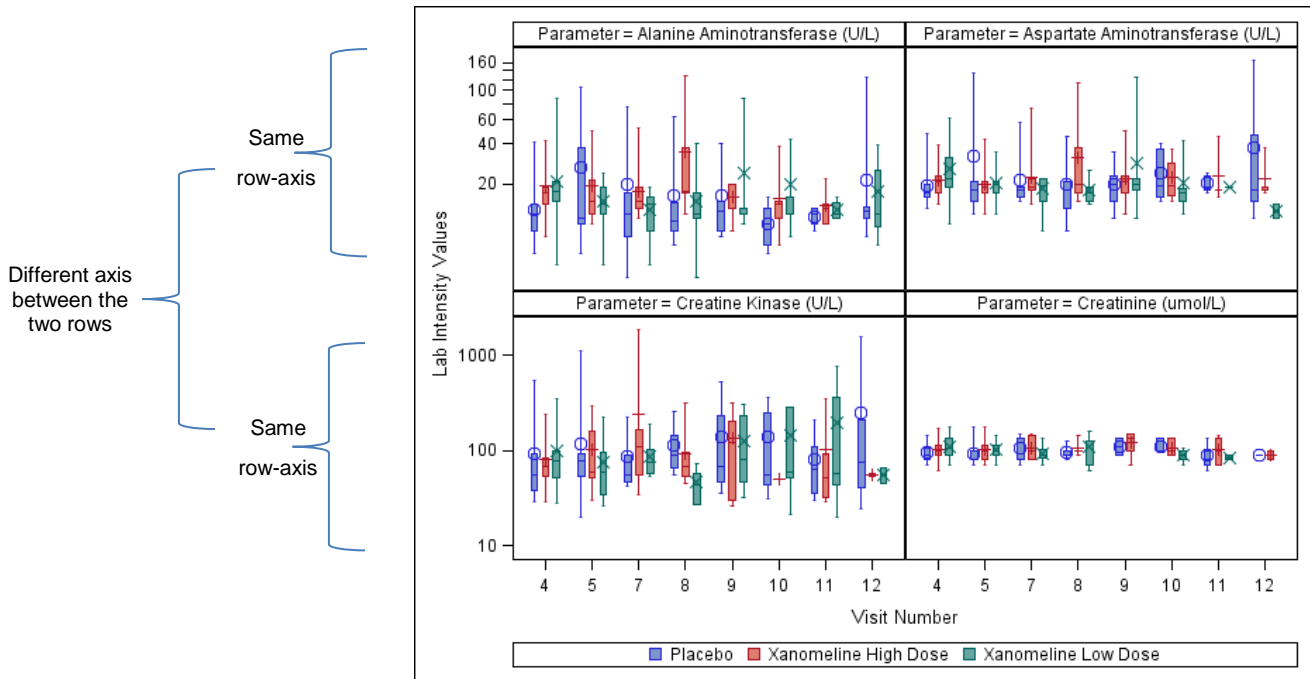
  layout prototype;
    boxplotparm x = visitnum y = coll stat = _NAME_ / group = trta
      groupdisplay = cluster name = "trtleq";
  endlayout;

  sidebar;
    discretelegend "trtleq";
  endsidebar;

  endlayout;
endgraph;

```

Figure 7: Boxplot with DATAPANEL



In Figure 7, you will notice that the columns have the same column-axis. In all the laboratory parameters the visit numbers range from 4 to 12. You will also notice that each row has the same row-axis. That is, the parameters, Alanine Aminotransferase (ALT) and Aspartate Aminotransferase (AST) range from approximately 1 to 160 and the parameters Creatine Kinase and Creatinine range from 10 to approximately 1000. The fact that at least the row or the column has to have the same axes makes the DATAPANEL layout very good for making comparisons between the rows or the columns or both. For example, it is very easy to notice that the variability in the Creatinine parameter is much smaller than the variability in Creatine Kinase, which is likely due to the difference in unit, and so a different axis might be better for a smaller unit.

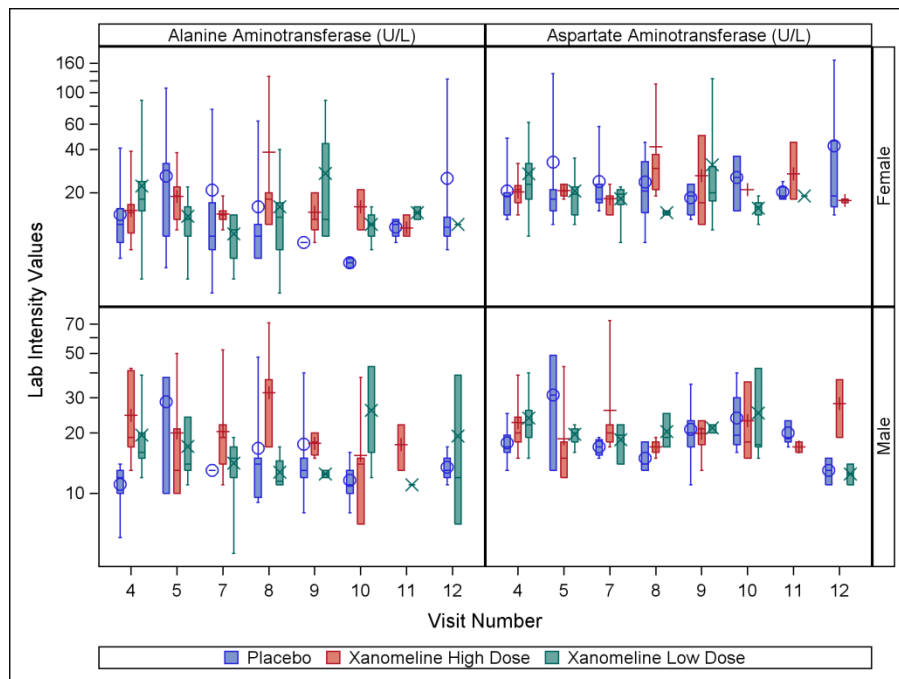
If you were interested in looking at data with different units or seeing how the data within each parameter is distributed, so that you can assess the treatment difference, or investigate if there are any points outside the limits then you might be interested in using a layout that has the functionality for using independent axes: the LATTICE layout. More details on how to use independent axes can be seen in (Harris, 2016, pp. 15-16).

DATALATTICE LAYOUT

The DATALATTICE layout is similar to the DATAPANEL layout. The main difference is that you can only exactly specify exactly two classification variables in the DATALATTICE layout, whilst in the DATAPANEL layout you can specify one or more classification. In the DATALATTICE layout, the values of one of your variables are shown in each column, and the values of your other variable are shown in the row. This layout is useful when you want to compare and contrast the values from two variables.

In Figure 8, you will notice that the laboratory test values are shown in the column headers and the gender of the subjects is shown in the row headers. The figure below is more for comparing the values of the two laboratory tests within female subjects, and within male subjects. If you wanted to compare males and females within each laboratory test, then you will swap the variables around that are used in the **rowvar** and **columnvar** options. The code used to plot Figure 8 is below.

Figure 8: Boxplot with DATALATTICE layout



```
begingraph;
  layout datalattice rowvar = sex columnvar = param /
  headerlabeldisplay=value columns = 2 rows = 2
  rowaxisopts=(type = log label = "Lab Intensity Values" rowdatarange=union;

  layout prototype;
    boxplotparm x = visitnum y = coll stat = _NAME_ / group = trta
    groupdisplay = cluster name = "trtleg";
  endlayout;

  sidebar;
    discretelegend "trtleg";
  endsidebar;

  endlayout;
endgraph;
```

CONCLUSION

GTL can be used to produce very sophisticated graphs. For a large amount of the graphs such as a single-cell basic plots, and fit and confidence plots the learning curve from SGPLOT to GTL is quite

minimal. For multi-cell categorization plots and distribution plots, the learning curve is slightly steeper but something that is worth doing because some graphs can only be produced with GTL, and also with GTL you can produce almost any graph that you would like.

For single-cell layouts you should use the OVERLAY layout, and for multi-cell layouts where you want to compare the graphs side by side then you should use the DATAPANEL layout. If you have exactly two classification variables that you want to compare then use the DATALATTICE layout. For all other multi-cell plots then you should use the most flexible layout: the LATTICE layout.

REFERENCES

- CDISC. (2013). CDISC. Retrieved June 2015, from SDTM/ADaM Pilot Project:
http://www.cdisc.org/system/files/members/article/application/zip/updated_pilot_submission_package.zip
- Harris, K. (2015). Picture this: Hands-on SAS Graphics Session. *PharmaSUG 2015* (pp. 6-7). Orlando: PharmaSUG.
- Matange, S. (2013). *Getting Started with the Graph Template Language in SAS: Examples, Tips, and Techniques for Creating Custom Graphs*. Cary, NC: SAS Institute Inc.
- McConville, K., & Much, K. (2015). Creating Sophisticated Graphics using Graph Template Language. *PharmaSUG 2015* (p. 1). Orlando: PharmaSUG

ACKNOWLEDGMENTS

I would like to thank Adrienne Bonwick for reviewing this paper.

RECOMMENDED READING

- Matange, S. (2013). *Getting Started with the Graph Template Language in SAS: Examples, Tips, and Techniques for Creating Custom Graphs*. Cary, NC: SAS Institute Inc.

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>

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

<OPTIONS1>

linearopts = (viewmin=60 viewmax=180 tickvaluesequence = (start=60 end=180 increment=20))