PhUSE EU Connect 2018

Paper PP20

Producing Panel by Dose plots using SGPANEL and SGRENDER

Rakesh Reddy, ICON plc, Marlow, UK

ABSTRACT

Graphs are a visual representation of the data. In a clinical trial setting, figures show the general trend and the relationships of variables collected during the conduct of a study. In the pharmacokinetic world, graphs play an important role in showing the drug's concentration profiles. The line graph is a typical way of presenting drug's concentration. However, depending on the data available and the analysis required, different visual presentations such as scatter plots, bar plots and many more, are needed to fully communicate the information at hand. Oftentimes, panel by dose plots are requested to plot clinical data such as vital signs and electrocardiogram (ECG) data against pharmacokinetic concentration data over time to spot existing relationships between the two parameters. These paneled graphs can be achieved by using PROC SGPANEL and PROC SGRENDER (using GTL). This paper will explore both the procedures for producing the panel by dose plots along with the pros and cons of different procedures.

INTRODUCTION

SAS/GRAPH® has been a standard in analytical reporting since the early days. Over the years it has improved in many ways. One of the recent improvements has been the addition of the Graphics Template Language (GTL). Based on the templates used for the Output Delivery System, the GTL gives users amazing control over the look and feel of the graphics that they create. However, the GTL is not necessarily for the faint of heart. To help people use this capability more easily, SAS has created the Statistical Graphics (SG) procedures: SGPLOT, SGPANEL, SGSCATTER, and SGRENDER. With thoughtful default values for color and layout, these procedures allow the user to create powerful graphics pretty easily.

SGPANEL PROCEDURE

The SGPANEL procedure is designed to produce a panel of plots based on the values of one or more classification variables. More specifically, SGPANEL procedure organizes plots into multiple panels which are used to compare various plots with regards to their classification variables. Classification variables can be designated as row or column variables, or there can be multiple classifications. Graphs are drawn for each combination of the levels of classification variables, showing a subset of the data in each cell. The syntax of SGPANEL is almost identical to SGPLOT procedure, so it is easy to convert SGPLOT code to SGPANEL to produce panel of plots by adding the PANELBY statement.

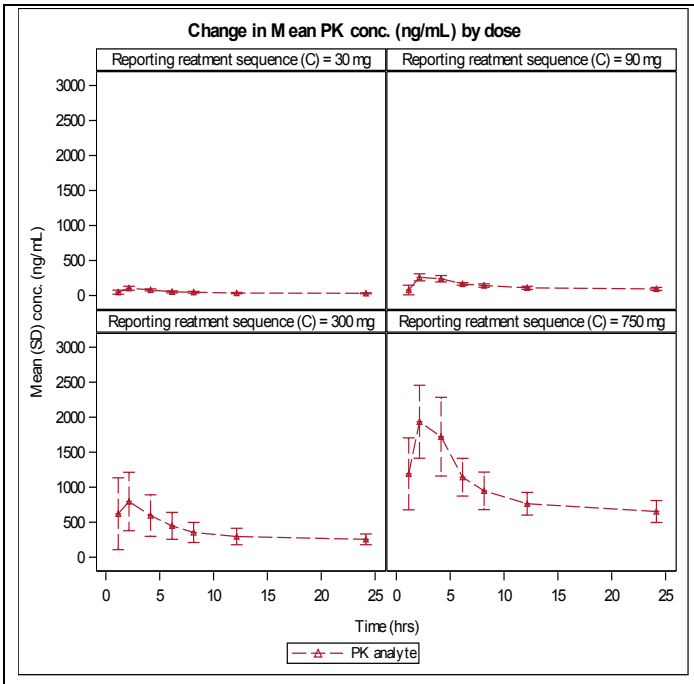
THE PANELBY STATEMENT

The PANELBY statement is required and must be the first statement before any plot, axis or legend statements. The procedure will use what you specify in the PANELBY statement to define the overall layout of the panel. The syntax for PANELBY is

```
PANELBY variables(s) </ option(s)>;
        option(s) can be one or more of the following:
        BORDER|NOBORDER
        COLHEADERPOS= TOP|BOTTOM|BOTH
        COLUMNS= n
        LAYOUT= LATTICE|PANEL|ROWLATTICE|COLUMNLATTICE
        MISSING
        NOVARNAME
        ONEPANEL
        ROWHEADERPOS= RIGHT|LEFT|BOTH
        ROWS= n
        SPACING= n
        SPARSE
        START= TOPLEFT|BOTTOMLEFT
        UNISCALE= ROW|ALL
```

In this paper we will not cover many of these options. Please refer to SAS documentation for more detail.

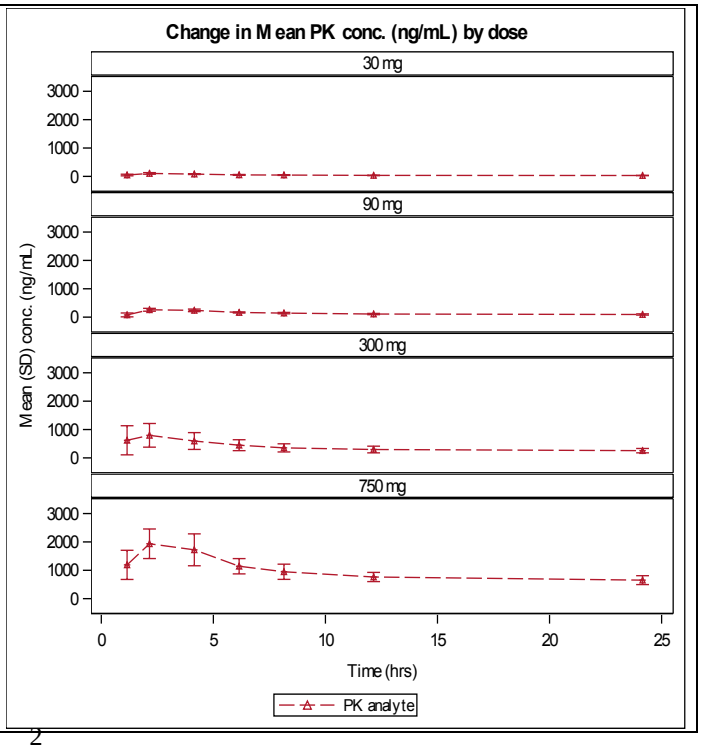
The following example shows the basic mean plot of PK concentration over time with Mean PK concentrations on the vertical axis, Time on the horizontal axis and the class variable REPTRTC (Treatment Dose) used to split the graph into different panels.




```
title1 "Change in Mean PK conc. (ng/mL) by dose";
proc sgpanel data=thisfig2 ;
panelby reptrtc; /*treatment dose*/
series  x=stm_pk y=mn_pk /
        markers
        lineattrs=GraphData2
        markerattrs=GraphData2(symbol=triangle)
        name="pk"
        legendlabel="PK analyte";
scatter x=stm_pk y=mn_pk /
        markerattrs=GraphData2
        errorbarattrs=GraphData2
        yerrorlower=low_pk
        yerrorupper=upp_pk;
run;
```

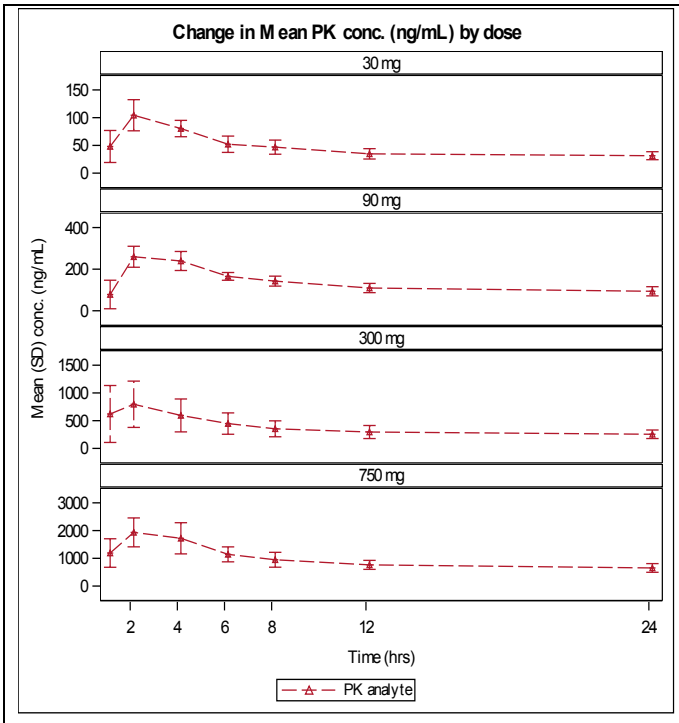
By default, the PANEL layout is used which supports any number of classification variables. One of the most important options to be used within the PANELBY statement is the LAYOUT. The two primary forms of the panel layout are LATTICE and PANEL (the default). This default layout arranges the cells by row from left to right and top to bottom with the headings placed at the top of each cell. This layout can be controlled with the START= option. The number of columns per row is determined by the number of levels of the grouping variable to create a cell. The layout can be controlled by the options, COLUMNS, ONEPANEL, and ROWS.

```
title1 " Change in Mean PK conc. (ng/mL) by dose ";
proc sgpanel data=thisfig2 ;
panelby reptrtc /  novarname layout=panel
                   columns=1 rows=4;
series  x=stm_pk y=mn_pk /
        markers
        lineattrs=GraphData2
        markerattrs=GraphData2(symbol=triangle)
        name="pk"
        legendlabel="PK analyte";
scatter x=stm_pk y=mn_pk /
        markerattrs=GraphData2
        errorbarattrs=GraphData2
        yerrorlower=low_pk
        yerrorupper=upp_pk;
run;
```



In the above example, we have added COLUMNS= and ROWS= options to control the number of columns and rows in the layout since we know that there are only 4 levels within the classification variable REPTRTC. However you don't have complete control with these options. The procedure will still make its own decisions about the layout if the settings don't fit well into the page. Also, these options have no effect if you use the ONEPANEL option and the LATTICE layout. Only one of them will have an effect if you use the ONEPANEL and the PANELBY layout. In that case, if you specify both COLUMNS= and ROWS=, then the COLUMNS= value will be used and the ROWS= value ignored. Also the NOVARNAME option is used to remove the variable name and the "=" symbol from the cell headings. This provides clean heading labels by avoiding the redundant information. If there are two PANELBY variables that have the same levels, however, then the variable name may be necessary.

The above figure still doesn't look complete, the error bars are not clearly visible and the y-axis is same across all the doses. But the y-axis that creates the plots is very different from each other since the concentrations depend on the dose received. By default, the SGPANEL procedure creates uniform row axes and uniform column axes across all graphs generated in a run. However, the PANELBY statement does contain an option called UNISCALE that enables you turn off the uniformity of either the row or column axes. By setting the uniform scale, each plot can have one axis with an independent range. In this case, we would like to have an independent y-axis for each dose and a uniform x-axis with the time points. Also the time points on the x-axis are uniformly distributed and we would like to have only specific time points on the x-axis, but only within the actual range in the data set. We can achieve this, by passing the list of values in the COLAXIS statement with the VALUESHINT option. The example plot below shows the change in Mean PK concentration plots by dose with an independent y-axis for each dose.




```
title1 " Change in Mean PK conc. (ng/mL) by dose ";
proc sgpanel data=thisfig2 ;
panelby reptrtc /  novarname layout=panel
                   columns=1 rows=4
                   uniscale=column;
series  x=stm_pk y=mn_pk /
        markers
        lineattrs=GraphData2
        markerattrs=GraphData2(symbol=triangle)
        name="pk"
        legendlabel="PK analyte";
scatter x=stm_pk y=mn_pk /
        markerattrs=GraphData2
        errorbarattrs=GraphData2
        yerrorlower=low_pk
        yerrorupper=upp_pk;

label mn_pk="Mean (SD) conc. (ng/mL)";
colaxis label="Time (hrs)"
        valueshint
        values=(0 1 2 4 6 8 12 24);
rowaxis thresholdmin=1 thresholdmax=1;

keylegend "pk";
run;
```

In summary, SGPANEL is quite a comprehensive and powerful procedure in the Statistical Graphics family to create paneled plots with various layouts. It allows the intuitive display of complex information, high-dimensional information. Knowing how to use the many options will help you customize your graphs to best present your data.

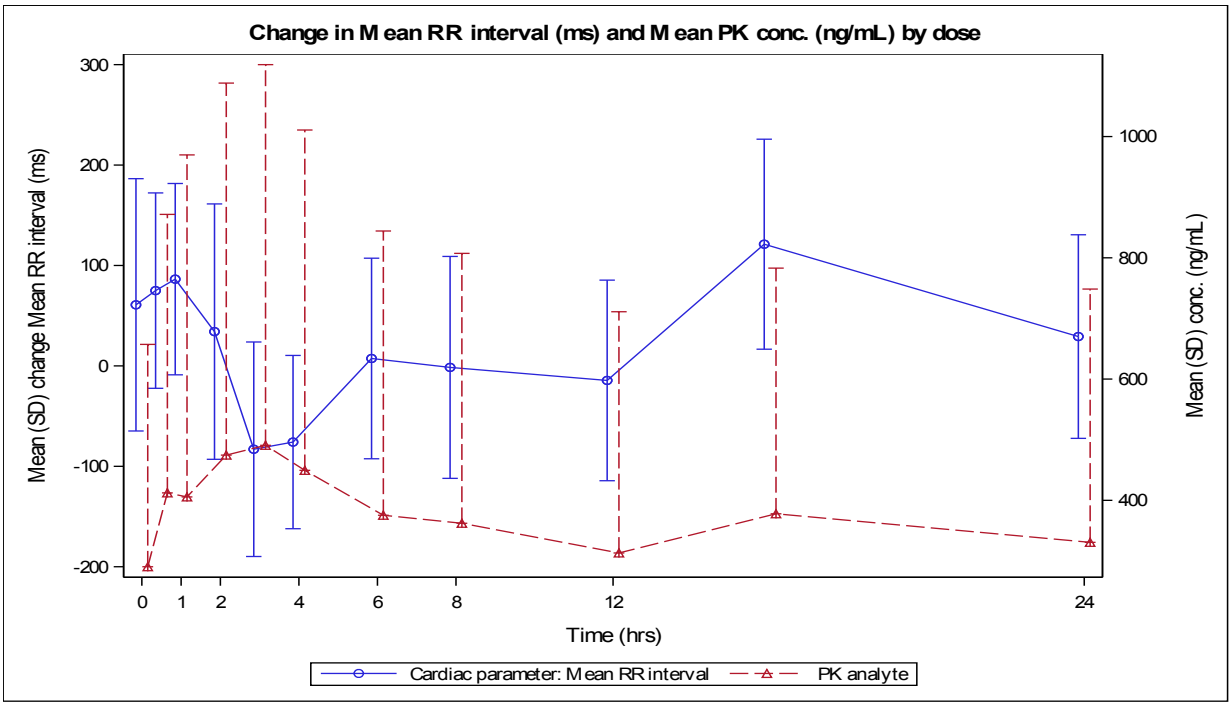
SGRENDER PROCEDURE

There may be times that you want to create a custom graph that you cannot completely make using the SGPANEL procedure. In those cases, using the Graph Template Language (GTL) may be your best option to design custom display and then use PROC SGRENDER to create the output by using that graph template and your data. GTL is the language used by the statistical procedures to create their displays by using the ODS Statistical Graphics Infrastructure. GTL enables you to create custom displays with a high degree of control and flexibility, but with that capability comes a higher degree of complexity than that of the "SG" procedure syntax. A discussion of GTL is beyond the scope of this paper (see Jeff Cartier's SUGI 31 paper, "A Programmers Introduction to the Graph Template Language" (2006)); however, if you use GTL to create custom displays, the SGRENDER procedure supports the following useful features:

- BY-group support – enables your templates to be rendered using BY-grouped data
- Special dynamics – dynamics such as _BYLINE_, _BYVAR_, _BYVAR2_, _BYVAL_, _BYVAL2_, _LIBNAME_, and _MEMNAME_ are surfaced through the procedure and are available for use in your template.
- Paging support – automatic paging support when a DATAPANEL or DATALATTICE layout is used in your template

There may be some users who are not sure how to create a GTL definition or would like to have a good starting point. Fortunately, the SGPLOT and SGSCATTER procedures have an option on the procedure statement called TMPLOUT that writes out a SAS file of the GTL definition for the graph you have in the procedure. You can then work this definition to add options not supported in the SG procedures or use this definition as a "building block" for a larger paneled graph.

For example, suppose you wanted to create the same paneled graph like above produced by SGPANEL procedure but to plot both the clinical data and pharmacokinetic concentration data over time on the same graph, then you can turn to GTL since the SGPANEL procedure doesn't support the second y-axis on the same plot. As a starting point, the below example figure is created by using SGPLOT procedure which plots both the parameters on the same graph, but for an overall, not by dose.

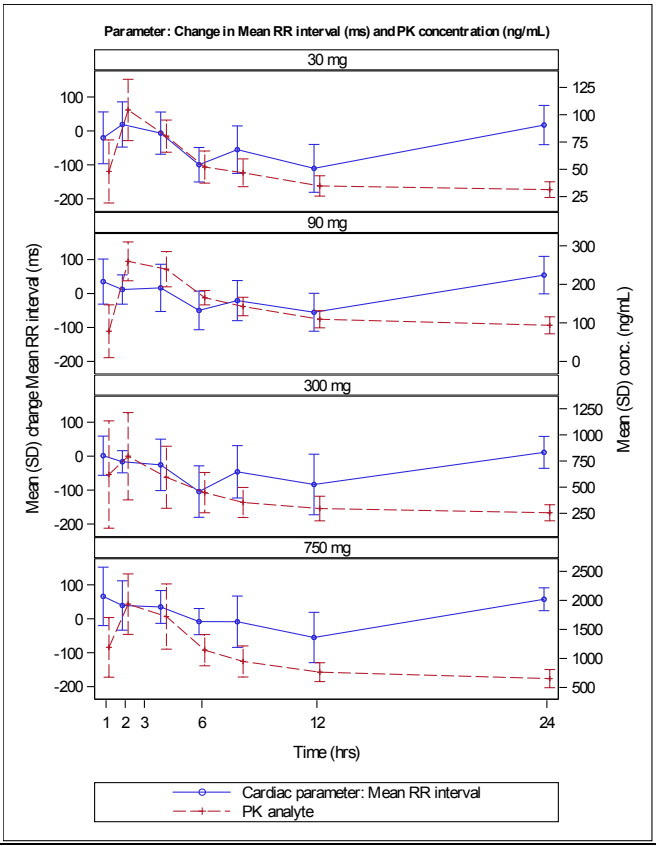



```
title1 "Change in Mean RR interval (ms) and Mean PK conc. (ng/mL) by dose";
proc sgplot data=thisfig2 tmplout='temp.txt';
  * plot mean change in ECG interval, left y-axis;
  series  x=stm_pd y=mn_pdc / markers lineattrs=GraphData1 markerattrs=GraphData1
                              name="ecgparm" legendlabel="Cardiac parameter: Mean RR interval";
  scatter x=stm_pd y=mn_pdc / markerattrs=GraphData1 errorbarattrs=GraphData1
                              yerrorlower=low_pdc yerrorupper=upp_pdc;
  label mn_pdc="Mean (SD) change Mean RR interval (ms)";
  * plot mean PK concentration over time, right y-axis;
   series  x=stm_pk y=mn_pk / y2axis markers lineattrs=GraphData2 markerattrs=GraphData2(symbol=triangle)
                              name="pk" legendlabel="PK analyte";
   scatter x=stm_pk y=mn_pk / y2axis markerattrs=GraphData2 errorbarattrs=GraphData2
                              yerrorlower=low_pk yerrorupper=upp_pk;
   label mn_pk="Mean (SD) conc. (ng/mL)";
   xaxis label="Time (hrs)" valueshint values=(0 1 2 4 6 8 12 24 48);
   yaxis thresholdmin=1 thresholdmax=1;
   keylegend "ecgparm" "pk";
run;
```

The “temp.txt” file created from the SGPLOT procedure contains the GTL definition for this plot, which looks like the following:

```
proc template;
define statgraph sgplot;
begingraph / collation=binary;
EntryTitle "Change in Mean RR interval (ms) and Mean PK conc. (ng/mL) by dose" /;
layout overlay / xaxisopts=( Label="Time (hrs)" type=linear linearopts=( tickvaluelist=( 0 1 2 4 6 8 12 24 48 ) ) )
                                    y2axisopts=(labelFitPolicy=Split) yaxisopts=( labelFitPolicy=Split type=linear
                                    linearopts=( thresholdmin=1 thresholdmax=1 ) )
                                    y2axisopts=(labelFitPolicy=Split);
  SeriesPlot X='stm_pd'n Y='mn_pdc'n / display=(markers) Markerattrs=GRAPHDATA1 Lineattrs=GRAPHDATA1
                                       LegendLabel="Cardiac parameter: Mean RR interval" NAME="ecgparm";
  ScatterPlot X='stm_pd'n Y='mn_pdc'n / subpixel=off primary=true Markerattrs=GRAPHDATA1
                                        YErrorUpper='upp_pdc'n YErrorLower='low_pdc'n
                                        ErrorBarAttrs=GRAPHDATA1 NAME="SCATTER"
                                        LegendLabel="Mean (SD) change Mean RR interval (ms)" ;
  SeriesPlot X='stm_pk'n Y='mn_pk'n / yaxis=y2 display=(markers)
                                      Markerattrs=GRAPHDATA2( Symbol=TRIANGLE)
                                      Lineattrs=GRAPHDATA2 LegendLabel="PK analyte" NAME="pk";
  ScatterPlot X='stm_pk'n Y='mn_pk'n / subpixel=off yaxis=y2 Markerattrs=GRAPHDATA2 YErrorUpper='upp_pk'n
                                       ErrorBarAttrs=GRAPHDATA2 LegendLabel="Mean (SD) conc. (ng/mL)"
                                       NAME="SCATTER1";
  DiscreteLegend "ecgparm" "pk" / Location=Outside;
endlayout;
endgraph;
end;
run;
```

Now that you have your plot definition, you can extract the necessary pieces to construct the paneled graph.




```
/*Choose the template and the input dataset*/
proc sgrender data=thisfig2_ template=mytemplate;
   by ecgparam ecgparamlab ecg_unit pooltrtc
        ccnunt1a;
   * dynamic entries for the template;
   dynamic _pdparmlbl="Mean (SD) change Mean
                       RR interval (ms)";

   format reptrt treatlbl.;
   label mn_pdc="Cardiac parameter: Mean RR
                 interval"
         mn_pk="PK analyte";
run;
```

TEMPLATE CODE

```
proc template;

*define graph template;
define statgraph mytemplate;
   *defines the values for dynamic variables used in sgrender procedure to specify required attributes;
   dynamic _byline_ _byval_ _byval2_ _byval3_ _byval4_ _byval5_ _byval6_
           _byval7_ _byval8_ _byval9_ _byval10_ _pdparmlbl;
  *defines macro variables;
   mvar rowrange;
   * graph definition;
   begingraph;

   * parameter title;
        entrytitle textattrs=(size=0.35cm) "Parameter: Change in " _byval2_ " (" _byval3_ ")
                                           and PK concentration (" _byval5_ ")";

        *graph layout: panel of multiple graphs;
        *datalattice layout creates a grid of plots based on one or more classification variables and a graphical
        prototype This requires a row classifier, a column classifier, or both;
        *This layout is one column with 4 graphs;
             layout datalattice rowvar=reptrt/*classification variable*/ /
                                              columnweight=proportional
                                              rows=4/*user preference*/
                                             skipemptycells=true
                                             headerlabeldisplay=value
                                             headerlabellocation=inside
                                             columnaxisopts=(label="Time (hrs)" type=linear
                                             linearopts=(tickvaluelist=(0 1 2 3 6 12 24)))
                                             rowdatarange=rowrange rowaxisopts=(label=_pdparmlbl)
                                             row2datarange=union
                                             row2axisopts=(label="Mean (SD) conc. (ng/mL)");
                  *graph prototype for each cell in the panel;
                  layout prototype;

                       *ECG Parameter;
                       scatterplot x=stm_pd y=mn_pdc / yaxis=y YErrorUpper=upp_pdc
                                    YErrorLower=low_pdc markerattrs=graphdata1
                                    errorbarattrs=graphdata1;
                       seriesplot  x=stm_pd y=mn_pdc / yaxis=y name='ecgparm' display=(markers)
                                    lineattrs=graphdata1 markerattrs=graphdata1;

                       *PK concentration;
                       scatterplot x=stm_pk y=mn_pk / yaxis=y2 YErrorUpper=upp_pk
                                    YErrorLower=low_pk markerattrs=graphdata2
                                    errorbarattrs=graphdata2;
                       seriesplot  x=stm_pk y=mn_pk / yaxis=y2 name='pk' display=(markers)
                                    lineattrs=graphdata2 markerattrs=graphdata2;
                  endlayout;

              * legend;
             sidebar;
                  discretelegend 'ecgparm' 'pk' / across=1 down=2 displayclipped=true;
              endsidebar;

             endlayout;

   endgraph;

end;

run;
```

CONCLUSION

Obviously, each approach has its own advantages and disadvantages, and which one to use depends on the situation and the kind of plot. Both these procedures can be used to produce very sophisticated graphs. For a large amount of the graphs such as 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.

REFERENCES

- **Matange, Sanjay 2008.** Introduction to the Graph Template Language. SAS Global Forum 2008 Proceedings.http://support.sas.com/resources/papers/proceedings10/234-2010.pdf
- **Heath, Dan 2008.** Effective Graphics Made Simple Using SAS/GRAPH SG Procedures. SAS Global Forum 2008 Proceedings. http://www2.sas.com/proceedings/forum2008/255-2008.pdf
- **Kincaid, Chuck (2010)**, SGPANEL: Telling the Story Better. Proceedings of the 2010 SAS Global Forum 2010 Proceedings. http://support.sas.com/resources/papers/proceedings10/234-2010.pdf
- SAS® support (https://support.sas.com/en/support-home.html)

RECOMMENDED READING

- Matange, Sanjay. "Graphically Speaking: Data Visualization with a Focus on ODS Graphics". Cary, NC: SAS Institute. http://blogs.sas.com/content/graphicallyspeaking/.
- SAS® 9.3 ODS Graphics Procedures Guide
- SAS® 9.3 Graph Template Language User's Guide
- SAS® 9.3 Graph Template Language Reference
- SAS/STAT® 9.3 User's Guide

CONTACT INFORMATION

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

Name: Rakesh Reddy
Company: ICON plc
Email: rakesh.reddy@docsglobal.com
Web: https://www.iconplc.com/