PhUSE US Connect 2018

Paper DV11

How ODS GRAPHICS INFRASTRUCTURE can be used in Statistical Analysis

Harivardhan Jampala, Chiltern - a Covance Company, Cary, NC, USA

ABSTRACT:
The SAS® System provides many tools for creating statistical graphics. Statistical graphics reveal patterns, differences, and uncertainties that are not clear in tabular output. This paper illustrates the functionality of the Output Delivery System(ODS) graphics infrastructure and how it helps programmers and statisticians to create simple and complex graphics easier. ODS Graphics makes it easy to create modern analytical graphs in SAS. The system has different components for creating graphs, each suitable for a different task, some of them are as follows:

Create graphs automatically from SAS procedures. No additional knowledge of SAS/Graph coding is required.

Create graphs using **ODS Graphics Designer**, an interactive application.

Edit graphs using **ODS Graphics Editor**.

Create graphs using **Statistical Graphics (SG) Procedures**.

Create graphs using the **Graph Template Language (GTL)**.

Add Tooltips to Graphs

Combine multiple graphs into a single ODS output file.

Graphics provoke questions that stimulate deeper investigation, they add visual clarity and rich content to reports and presentations. Many SAS procedures that utilize ODS Graphics can produce graphs along with the relevant statistics they present. This enables the user to generate graphical integrated output from analytical procedures with virtually no additional effort to create a graph.

INTRODUCTION:

This paper is aimed at a casual user of the Output Delivery System (ODS) who is not familiar with new graph techniques available in SAS version 9.4 The Output Delivery System has many features that are well known, beyond the basic techniques, there are some essential features that I consider to be powerful capabilities of ODS Graphics. The purpose of this paper is to describe these features and to share some code and the reasons these features were employed.

Graphs play an important role in clinical research, SAS ODS Graphics, sometimes called SAS ODS Statistical Graphics is a SAS feature that can be used to create a wide variety of graphs. Good graphics are indispensable for modern data exploration and presentation, and to do this we often use SAS/Graphs. This paper introduces ODS Graphics for SAS users who are just starting to explore ODS Graphics. It aims to show that high-quality statistical graphs can be produced with very little programming code. All the graphs dealt with in this paper require minimal effort from the programmer when compared to traditional SAS/Graph procedures. It presents examples of SAS Graph procedures and Graph Template Language with some commonly used graph types and ways to customize them. A section of this paper illustrates the use of ODS Graphics Designer and ODS Graphics Editor to create figure shells with minimal editing of program code, this is very useful for statisticians and data managers. If a large amount of effort is required to create plots needed for analysis, it is worth the time invested in utilizing ODS Graphics Infrastructure.

Create Graphs Automatically from SAS procedures. No Additional Knowledge of SAS/Graph Coding is Required.

SAS ODS Graphics is an extension of the SAS Output Delivery System (ODS). ODS manages all output that is created by procedures and enables you to display the output in a variety of formats like HTML, RTF and PDF.

Many SAS analytical procedures that support ODS Graphics functionality can produce graphs as automatically as they produce the relevant statistics.

Proc Freq is a common procedure among users. Using the plots options in the table statement creates graphs that can be used for graphical presentations.

Figure 1 below shows that adding /plots =freqplot to the tables statement creates a simple count graph.
Figure 2 below shows that adding /plots =freqplot/(twoway = grouphorizontal) to the tables statement creates a grouped graph.

FIGURE 1: Simple plot from PROC FREQ using /plots=freqplot option

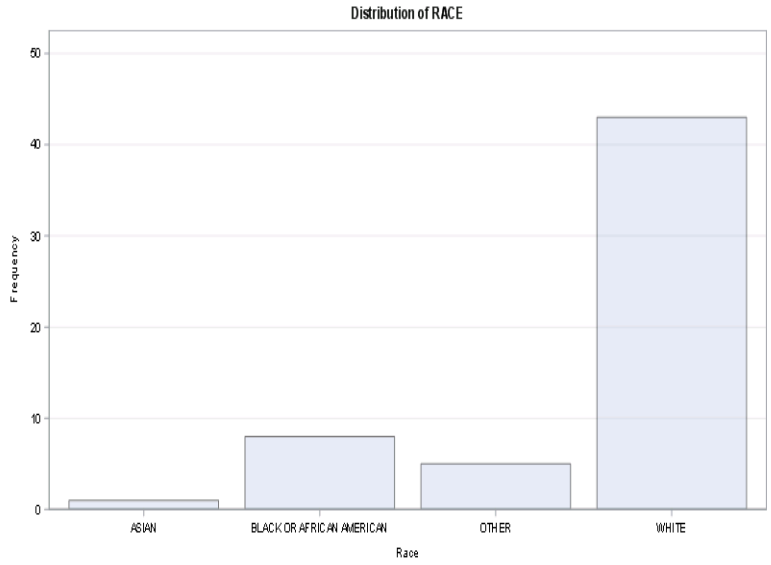


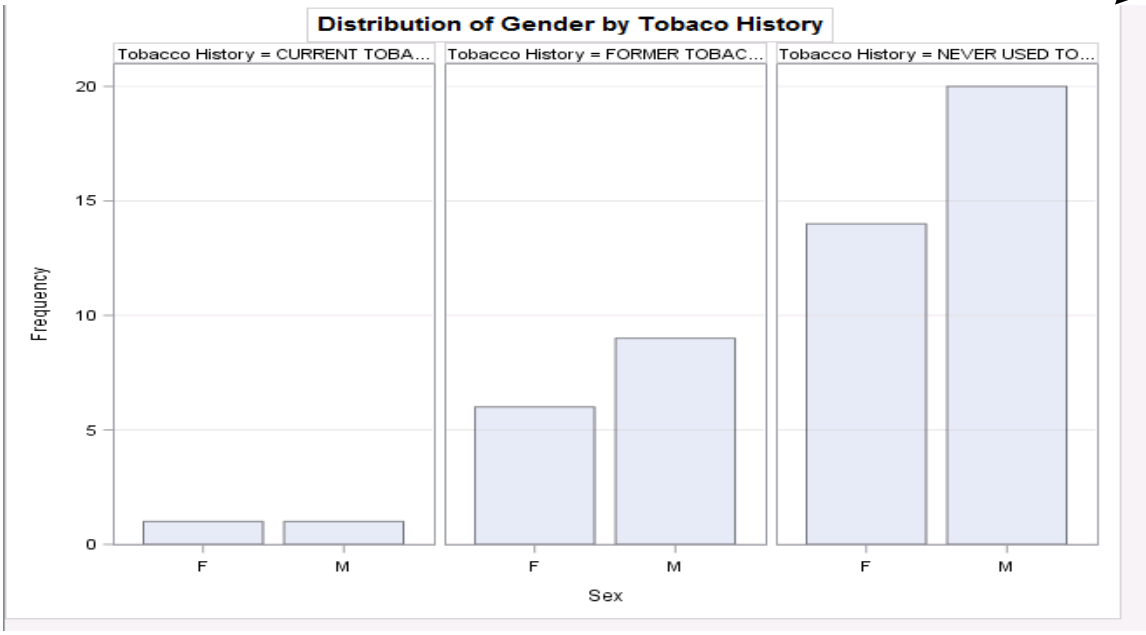
FIGURE 1: Code

```
ods graphics on;
proc freq data=test.adsl;
tables race/plots=freqplot;
run;
ods graphics off;
```

FIGURE 2: Code

```
ods graphics on;
proc freq data = test.adsl;
tables sex * SUNCFTOB /
plots (only) = freqplot/(twoway =
grouphorizontal);
run;
ods graphics off;
```

FIGURE 2: Groups of frequency using freqplot/(twoway = grouphorizontal) option.

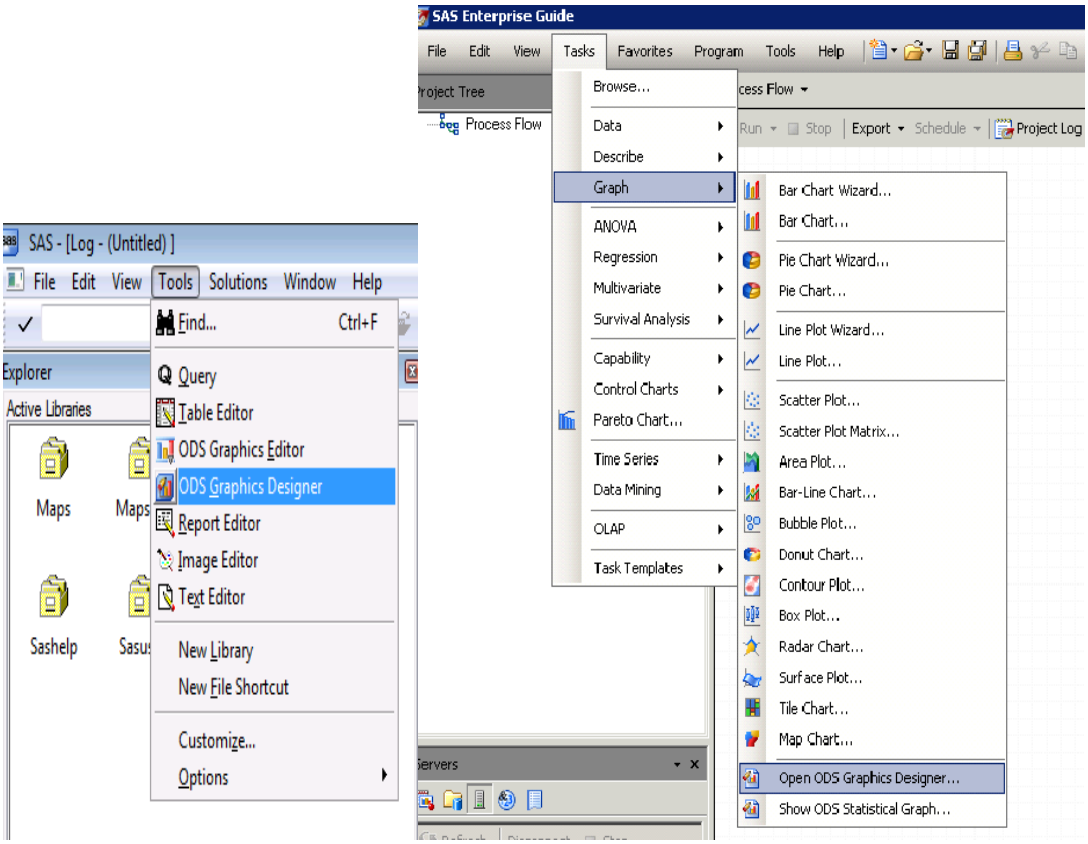


Create Graphs Using ODS Graphics Designer, an Interactive Application.

The ODS Graphics Designer provides an interactive interface for creating the same types of graphs produced by the procedures SGPLOT, SGPANEL, and SGSCATTER. It also allows the creation of some types of graphs that cannot be created with the above procedures.

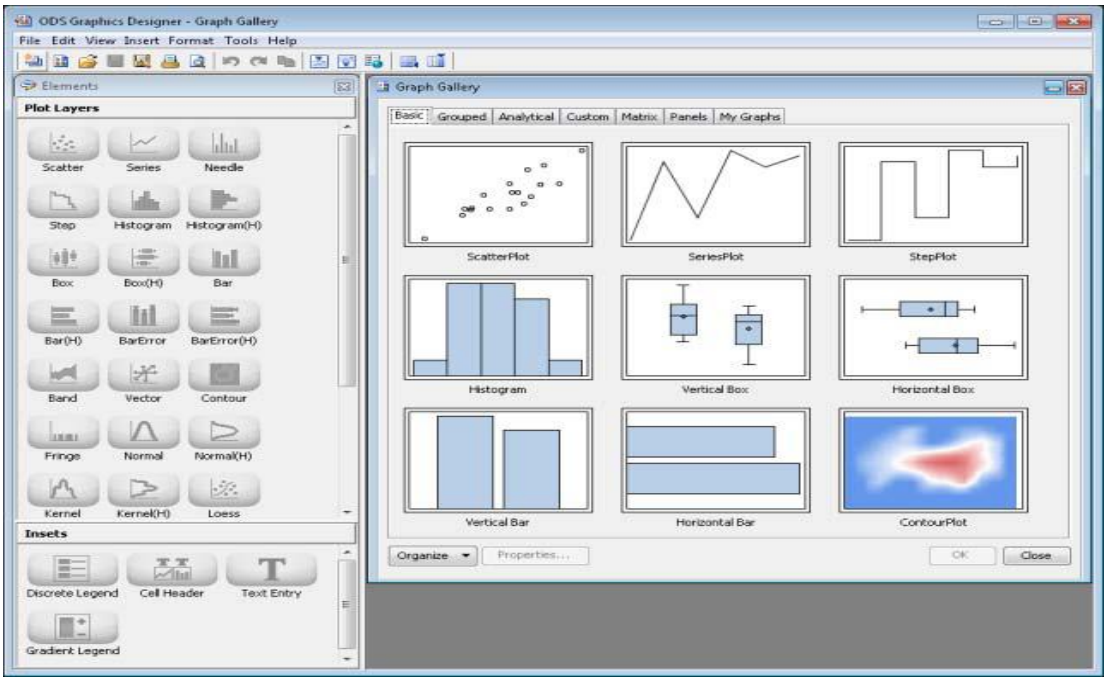
To invoke the ODS Graphics Designer, submit the following statement from your SAS Editor:
%sgdesign;

ODS GRAPHICS DESIGNER: The ODS graphics designer is generally used by analysts, statisticians, managers, academics, and enthusiasts who want to graphically explore data or present the results of their analyses. Using a point-and-click interface, users can create simple or complex graphical view of data for presentation without having to write code in Graph Template Language or SAS/Graph procedures.



DISPLAY 1: Navigating to initiate ODS Graphics designer interactively using PC-SAS or Enterprise Guide.

ODS GRAPHICS is a self-explanatory tool. Select a graph to be plotted and follow the prompts to create a simple graph or multiple plots in the same graph.



DISPLAY 2: ODS Graphics Designer Interface.

Edit Graphs Using ODS Graphics Editor:

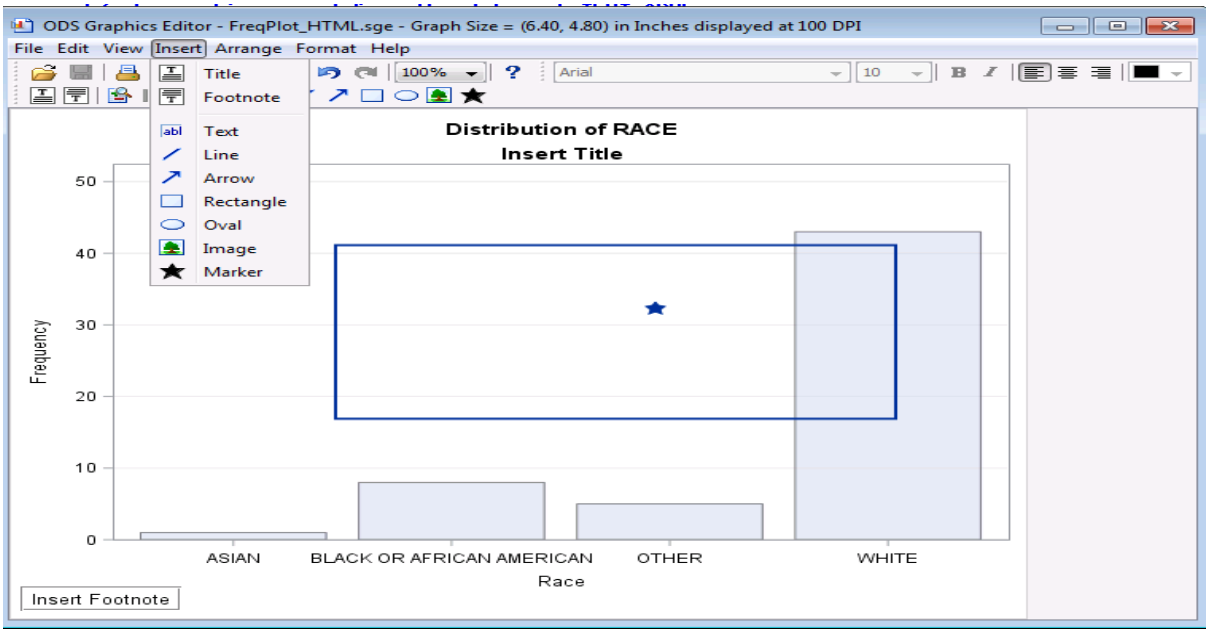
ODS Graphics Editor provides a graphical, interactive editor that enables you to modify various elements of a graph such as titles, lines, and markers. Users can annotate graphs by adding arrows, text, images, and other items. The editor works with editable graphs (SGE files). Users can also open and annotate PNG (Portable Network Graphics) files.

How to Edit Your Graphics Output:

Start by adding the **SGE=ON** option in the ODS statement in your SAS code before a procedure so that you can edit your graphics output:

```
ods listing sge=on;

proc freq data=test.adsl;
 tables race/plots=freqplot;
run;
```

The graphical output is saved as an SGE file in the **~/Projects/Files** directory. In the **Folders** section of the navigation pane, expand the Files folder and open the editable graph.



DISPLAY 3: ODS Graphics Editor.

Mock Shells can be created easily by a Statistician or a Programmer. The ODS Graphics Designer and ODS Graphics Editor are useful for creating figure shells without the need to spend a lot of time in SAS/Graph code.

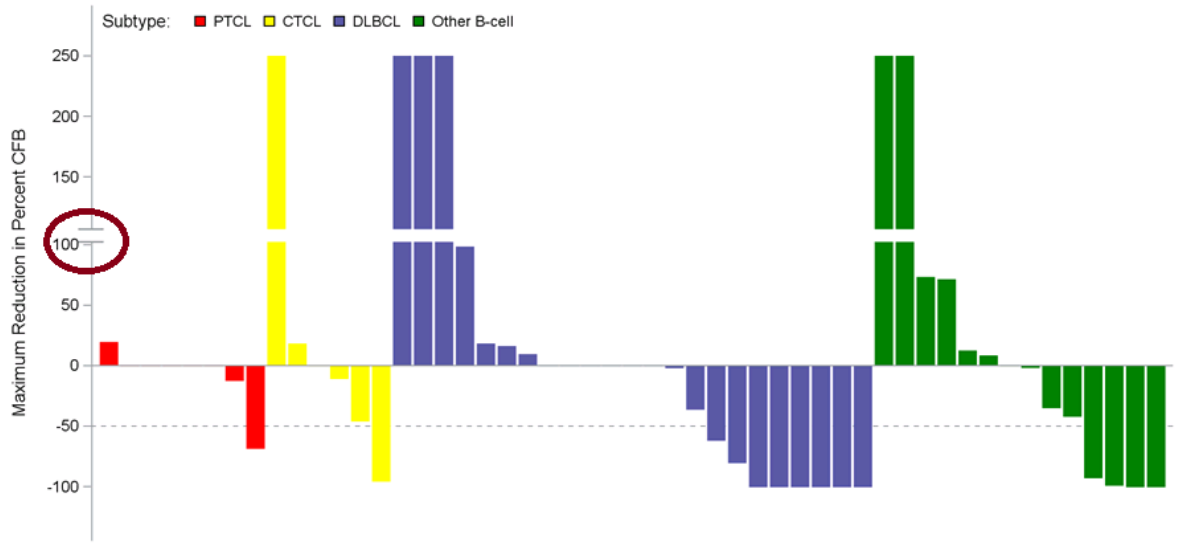
Create Graphs Using Statistical Graphics (SG) Procedures:

The three SG Procedures SGplot, SGPanel, and SGScatter are most commonly used these days. There are several types of graphs created using HBAR/VBAR, HBOX/VBOX, HLINE/VLINE, NEEDLE, SERIES, STEP etc., this paper does not deal with them in detail. Some scenarios where the user can employ simple methods to accomplish traditionally difficult tasks instead of elaborate SAS/Graph code are shown below.

Scenario #1: Waterfall Plot with break on y-axis indicating disease progression.

In this scenario, a waterfall plot is needed for a journal of clinical oncology. The graph must show the maximum percent change from baseline in total tumor burden by subtype without showing any of the X-axis values. A further request is that we show a break at y-axis = 100 because 100% is the cutoff point for Disease Progression (PD) for this specific tumor burden. This allows the reader to see the difference in tumor subtypes with more than 100% reduction. The challenge here is to create such a graph without resorting to the use of graph annotation and keep the process simple.

Use AXISBREAK= and AXISEXTENT options in the PROC SGPLOT options to break the axis, the RANGES= option in the YAXIS statement of PROC SGPLOT is used to specify the values at which cutoff points need to occur on the Y axis.



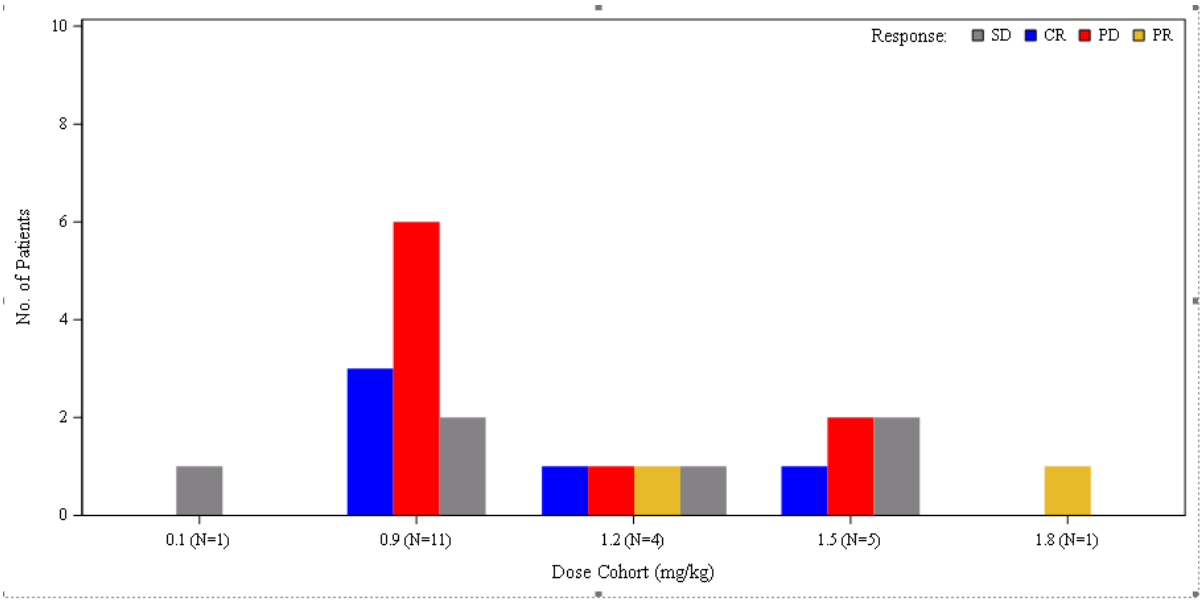
PLOT Scenario #1: Waterfall Plot with break on y-axis indicating disease progression

```
proc sgplot data=test pad=(bottom=8pct);
       styleattrs datacolors=(red yellow lib green) AXISBREAK=BRACKET axisextent=full ;
       refline -50 / axis=y lineattrs=(pattern=shortdash);
       vbar n / response=pcfb group=var  nooutline name="sty" legendlabel="Subtype";
       yaxis ranges=(-100-100 110-250) values=(-100 to 250 by 50) valueshint offsetmin=0.1 offsetmax=0.095;
     xaxis display=none  ;
   keylegend "sty"  / title="Subtype:" position = topleft location=inside noborder;
run;
```

Scenario #2: Cluster Groups based on response types

A graph is needed to display best overall response by dose cohort. The requirement is a cluster bar showing number of patients by cohort and response. This can be easily shown using the **response** and **group** and **groupdisplay** options as shown in the code below.

```
proc sgplot data=test ;
      styleattrs datacolors=(grey blue red BIOY) ;
      vbar trt / response=cntvar
        group=avalc groupdisplay=cluster  stat=mean
        baselineattrs=(thickness=0) name="sty" legendlabel="Response" nooutline;
     yaxis  values=(0 to 10 by 2);
    xaxis ;
     keylegend "sty"  / title="Response:" position = topright location=inside noborder;
 run;
```

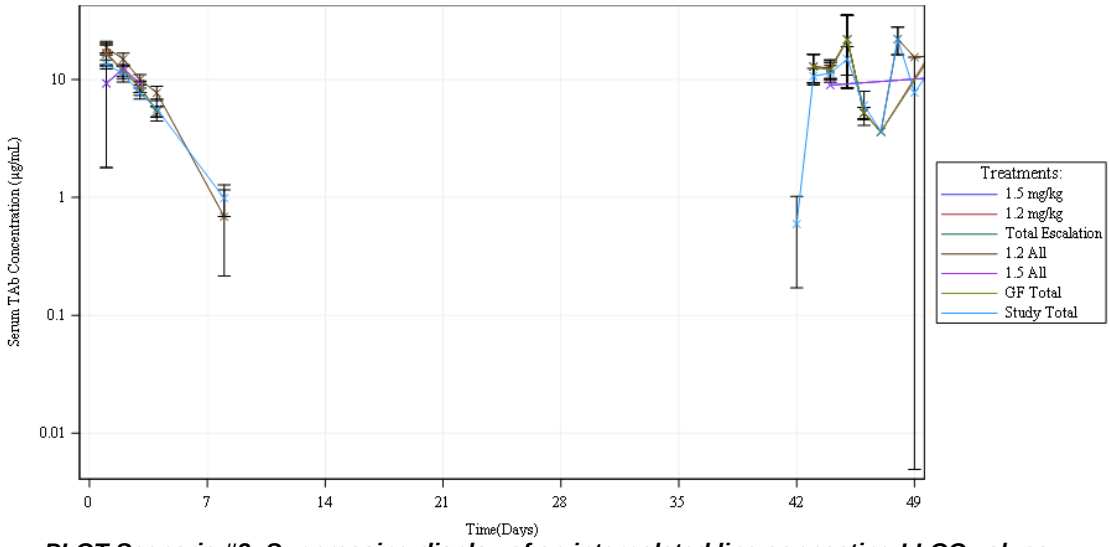


PLOT Scenario #2: Plot of Cluster Groups based on response types

Scenario #3: Suppressing display of an interpolated line connecting LLOQ values

A graph is needed for a pharmacokinetic figure of serum concentration-time profiles with error bars representing the standard deviation: The Concentration values between Day 7 and Day 42 days were below the Lower Limit of Quantification. The instructions are that LLOQ values must not be plotted and they must not be treated as missing for interpolations. The usual approach to handling LLOQ is to replace them with zero or with missing. Suppressing the display of the interpolated line connecting the Day 7 data points and the Day 42 data points with such an approach would not work for creating this figure. The solution is to use the **BREAK** option as shown below.

```
proc sgplot data=fin ;

xaxis grid values=(xval) label="Time(Days)" labelattrs=(size=9);
yaxis grid type = log label="&yax"  labelattrs=(size=9);
scatter x=ady y=mean / group=trtp markerattrs=(symbol=x) yerrorlower=sel yerrorupper=seu
        errorbarattrs=(color=black);
series x=ady y=mean / group=trtp BREAK  lineattrs=(pattern=solid) name="trtp" legendlabel="Treatments";
keylegend "trtp"  / title="Treatments:" position = right location=outside border;

run;
```

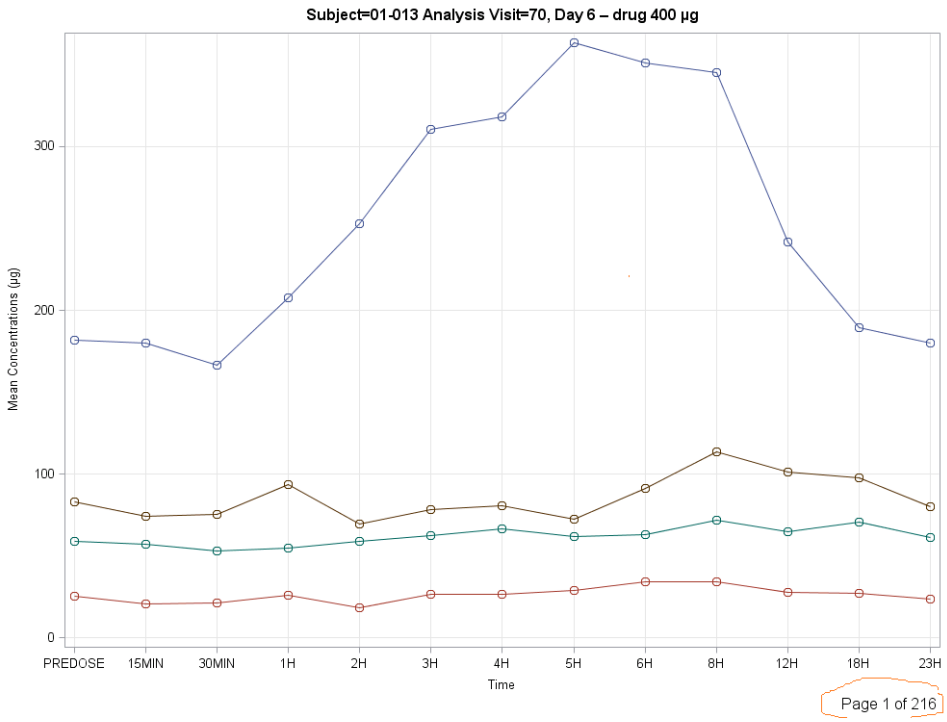


PLOT Scenario #3: Suppressing display of an interpolated line connecting LLOQ values

Scenario #4: Maintaining a dynamic x-axis on multiple plots with varying timepoints.

A graph is needed for a plasma concentration-time profile by treatment. The requirement is to draw a scatter and series plot for each subject and visit and drug dose. The challenge here is to keep the x-axis timepoints dynamic as some subjects might miss some time points. Using the **type=discrete** option in the xaxis statement to create a dynamic x-axis based on available timepoint data for that subject. Also note the footnote has a dynamic page X of Y update with each subject. This is done by utilizing **the #byval** option in the title or footnote statement as needed.

```
Title1 h=8pt  j=c font="simplex" "Subject = #byval1 and Analysis Day = #byval3  ";

footnote3 j=r "Page #byval4 of &totpg";

proc sgplot data=fin noautolegend;
   by usubjid avisitn avisit var;
         format atptn tpt. ;
         xaxis grid label="Time" type = discrete tickvalueformat = tpt. labelattrs=(size=9);

         yaxis label="&yax"  labelattrs=(size=9);
         scatter x=atptn y=mean /group=paramcd markercharattrs = mark markerattrs=(size=9)
                  yerrorlower=sel yerrorupper=seu;
      series x=atptn y=mean / group=paramcd lineattrs=(pattern=solid) ;
run;
```



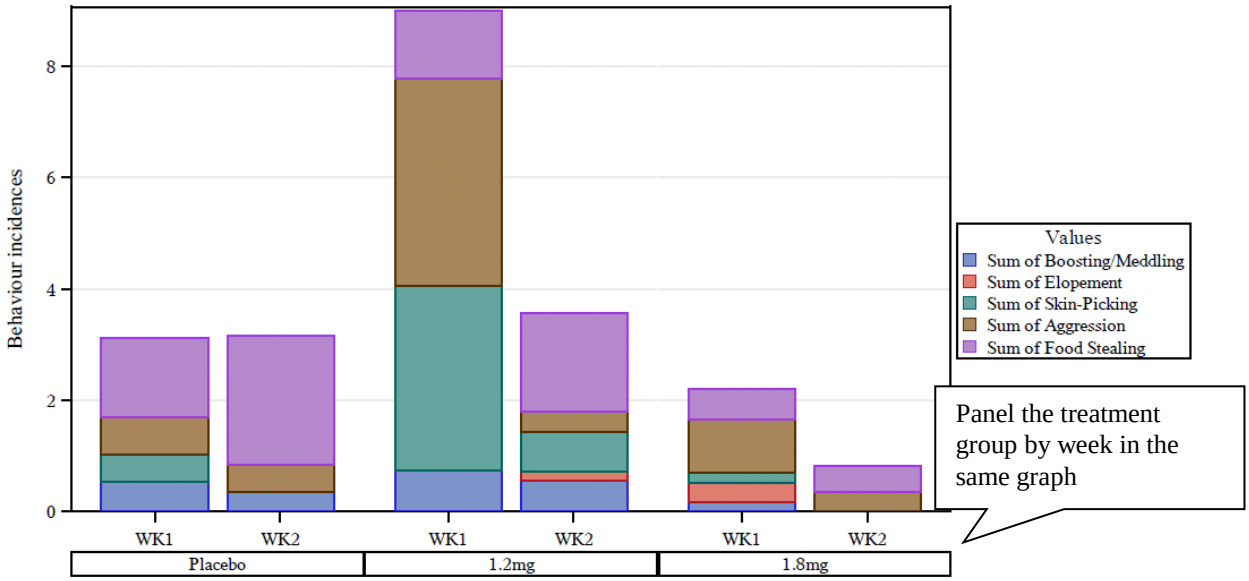
Scenario #4: Maintaining a dynamic x-axis on multiple plots with varying timepoints

Scenario #5: Combine multiple plots in the same panel of a bar chart

A graph is needed to plot behavior incidence per week for each behavior category from a daily journal.
The requirement is to show all the behaviors in a single bar by week within the treatment in one figure. In Proc SGPANEL, the PANELBY statement with a LAYOUT=COLUMNLATTICE option will present this display as shown below.

```
proc sgpanel data = figout_pl;
      format behavior $behav. avisitnum vistfmt. trtd trt.;
       panelby  trtd / layout=columnlattice onepanel novarname noborder colheaderpos=bottom;
       vbar avisitnum / response=dgorres group=behavior grouporder=data;
      colaxis  display=(nolabel) discreteorder=data;
      rowaxis /*values=(0 to 60 by 10)*/ grid;
```

```
      keylegend / across=1 position=right title="Values" ;
run;
```

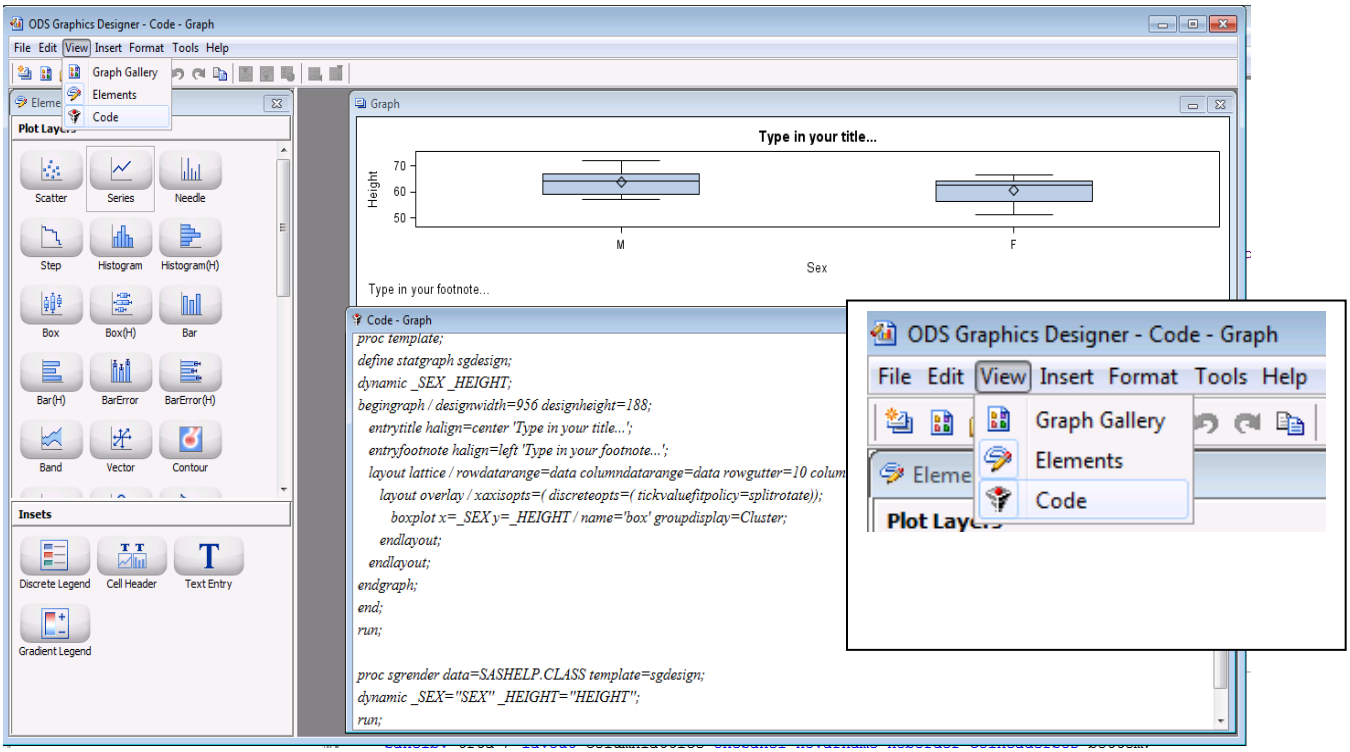


PLOT Scenario #5: Combine multiple plots in the same panel of a bar chart.

Complex Graphs Using the Graph Template Language (GTL):

A graph can be created without writing a single line of code. Using ODS graphics designer the user can create a graph interactively, the code for what has been designed will be accessible and editable. This code can be modified and customized to suit our requirements. It can also be exported to a SAS program to create a standalone program.

The user can access the SAS code for the graph after the interactive design process is complete by navigating to *View → Code* in the ODS Graphic Designer.

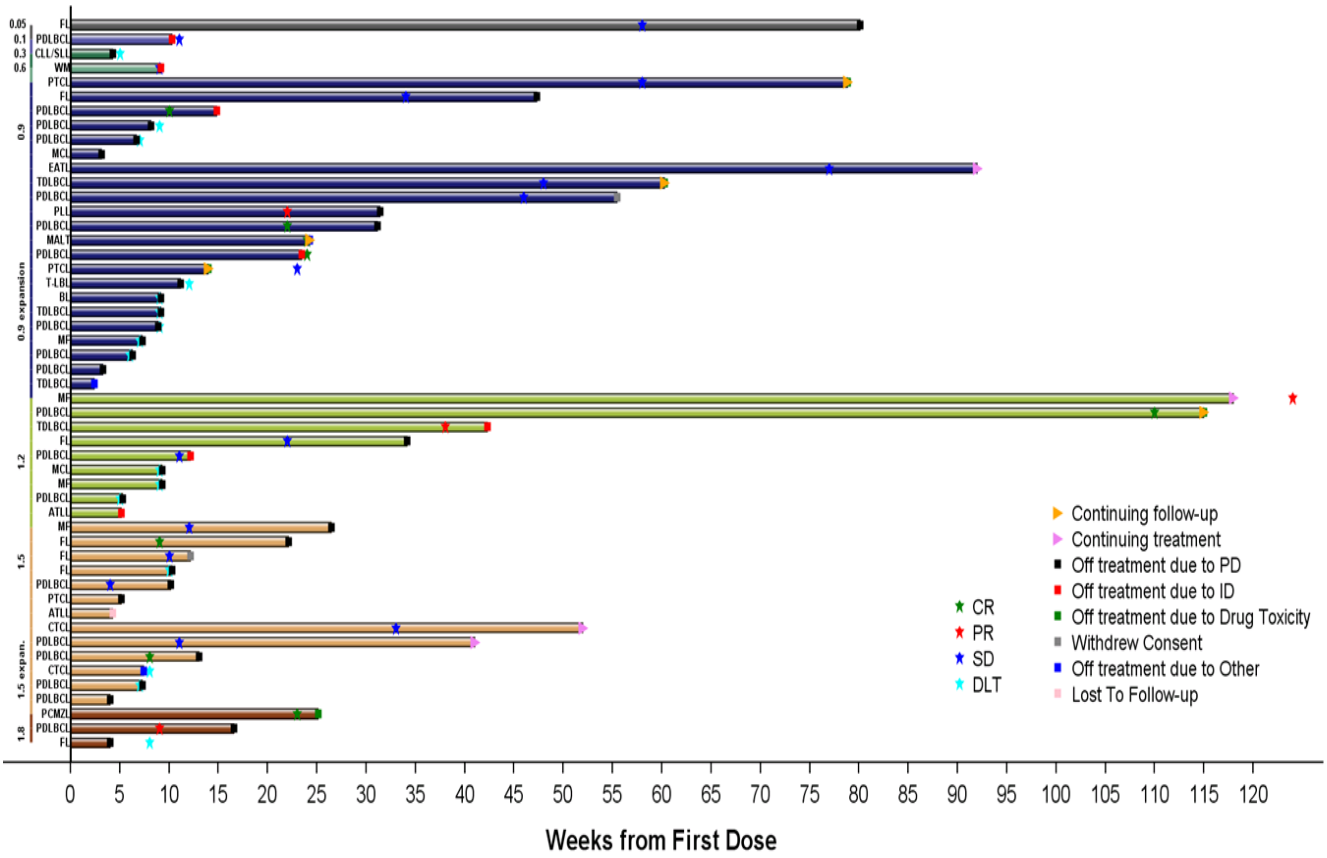


DISPLAY 4: ODS Graphics Designer Interface to display SAS code.

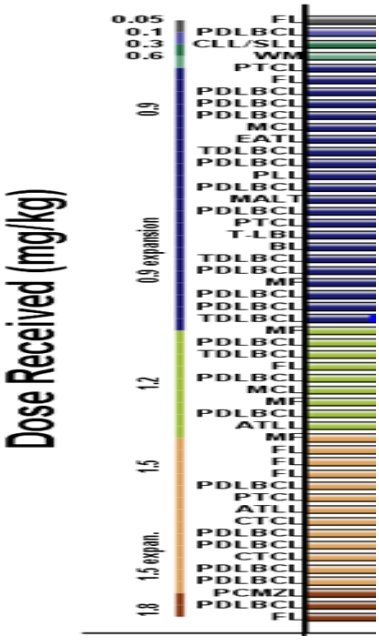
Here are some useful tips that can be used in Graph Template Language (GTL) without going into the details of the code. You can create complex graphics by taking the basic code generated by ODS Graphics Designer and modifying it. Below are a few examples with useful tips on creating complex graphs.

Example #1: Swimmer Plot (Patient Profile Graph) using TEXTPLOT Facility

In this example, the user can create a swimmer plot with duration of treatment, endpoints, disposition, tumour type, and treatment group per subject. Swimmer plots can be created in multiple ways but as you can tell from all the components in this graph, it can be quite complicated to program and sometimes requires the capabilities of GTL. In the plot below, the subjects are ordered by dose group, and the colour of the bars identifies the dose group of each subject. There is a requirement to add a specialized legend that is made up of a multicolour vertical line to the left of the y-axis with the colours segmented at the min and max y-axis values per dose group. The legend labels (treatment group) appear to the left of the vertical line but had to be oriented at -90 degrees if there is more than one subject in the treatment group and at 0 degrees if there is only one subject. In addition to the legend, the tumour type for each bar (subject) should appear just to the left of the y-axis. The vertical line can be created easily using the annotation facility, the required text can be plotted using the Textplot facility. Textplot is very similar to Scatterplot except that text values are plotted instead of symbols. Three separate textbox plot statements are used for the -90-degree legend labels, the 0-degree legend labels, and the tumour type values.



PLOT Example #1: Swimmer plot with duration of treatment

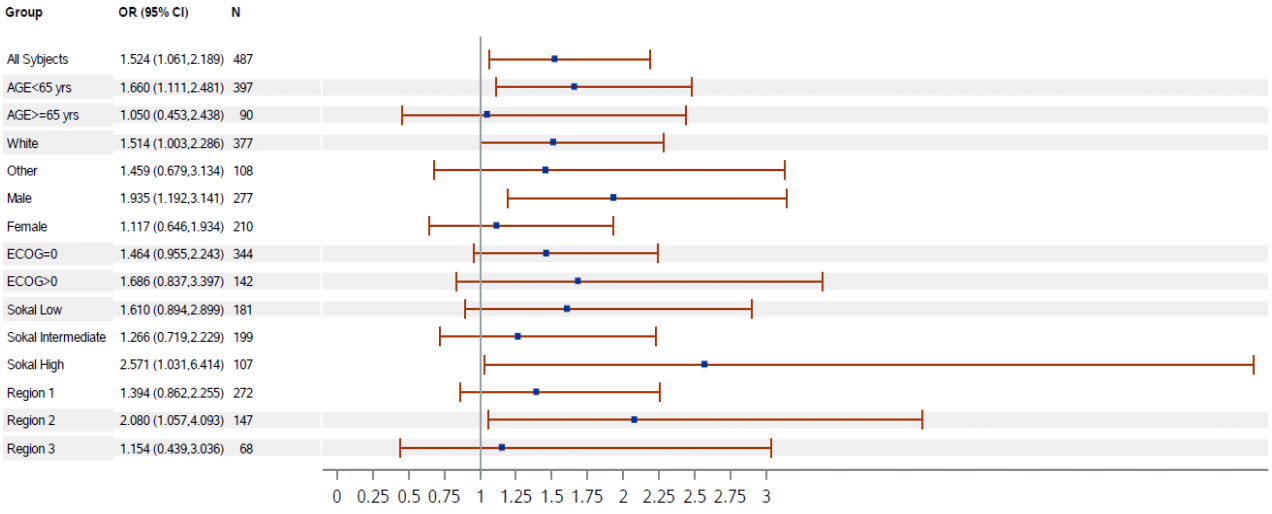


PLOT: Example #1 : Swimmer plot Y-axis zoomed

```
TEXTPLOT x=tp_x_0 y=yvar text=subtype /
        name="TP_1" textattrs=(color=black family="arial" weight=bold size=4) rotate=0 position=left;

TEXTPLOT x=tp_x_neg90 y=yvar text=trtylab_90 /
        name="TP_2" textattrs=(color=black family="arial" weight=bold size=4) rotate=90 position=right;

TEXTPLOT x=tp_x_neg y=yvar text=trtylab_0 /
        name="TP_3" textattrs=(color=black family="arial" weight=bold size=4) rotate=0 position=left;
```

Example #2: Dynamically changing colors without modifying proc template code. Assigning different colors to various elements in a forest plot.

Forest plots are easy to create using GTL when compared to using the SAS Annotate facility. The requirements might request a need to use different colors for various elements of a graph like background color or reference line color that can be altered without modifying the Proc Template code



PLOT Example #2: Assigning different colors to various elements in a forest plot.

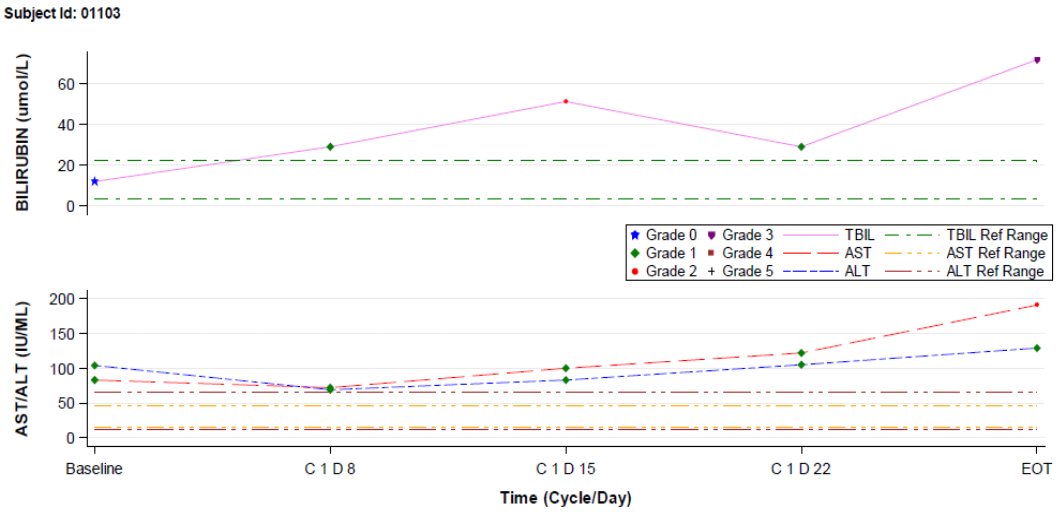
The portions of code used to alter the specific requirements for a forest plot can be dynamically updated using the DYNAMIC statement. The DYNAMIC statement creates a macro like variable to specify color modifications. The variables that determine the color and background color in proc template are supplied from Proc SGRender.

```
proc template;
  define statgraph Forest_HighLow/store=temp;
    dynamic  _color _backgrcolor;
    begingraph;
    layout lattice / columns=4 columnweights=(0.06 0.06 0.05 0.50);
                     entrytitle  "Plot of Odds Ratio"/textattrs=(size=8) ;
                     entrytitle halign=left " "/textattrs=(size=8) ;
                     entrytitle halign=left " "/textattrs=(size=8) ;

/*--Column headers--*/
       sidebar / align=top;
            layout lattice / rows=2 columns=4 columnweights=(0.06 0.06 0.05 0.50)
            backgroundcolor=_backgrcolor opaque=true;
            entry textattrs=(size=7 weight=bold) halign=left "Group";
                                 entry textattrs=(size=7 weight=bold)  halign=left "OR (95% CI)";
                                 entry textattrs=(size=7 weight=bold)  halign=left "N";
             endlayout;
       endsidebar;

/*--First Covariates column*/
            layout overlay / walldisplay=none xaxisopts=(display=none)
             yaxisopts=(reverse=true display=none tickvalueattrs=(weight=bold));
             referenceline y=ref / lineattrs=(thickness=15 color=_color);
             axistable y=subgroup value=subgroup / textgroup=type display=(values);
            endlayout;

                        .....>>>>>>>.;
endlayout;
endgraph;
end;
run;

proc sgrender data=Forest template=Forest_HighLow;
  dynamic _color='cxf0f0f0' __backgrcolor='white';
run;
```

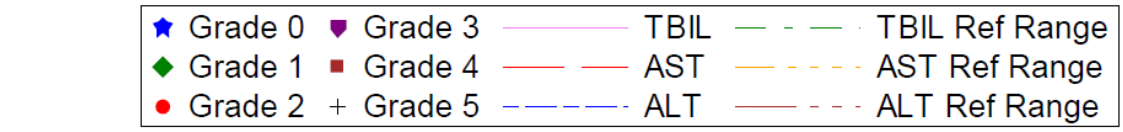
Example #3: Interpolating a common legend on a common x-axis for comparing multiple plots.

In this example a few Blood Chemistry parameters like Total Bilirubin, AST and ALT for a subject are combined into single plot with a common legend and x-axis. The requirement is to have only one x-axis and each subject should be plotted in a single page to visually compare the AST/ALT results versus the Bilirubin results.

This can be achieved by creating a lattice layout with 3 separate blocks of layouts in the Proc Template code.



PLOT Example #3: Interpolating a common legend on a common x-axis for comparing multiple plots.

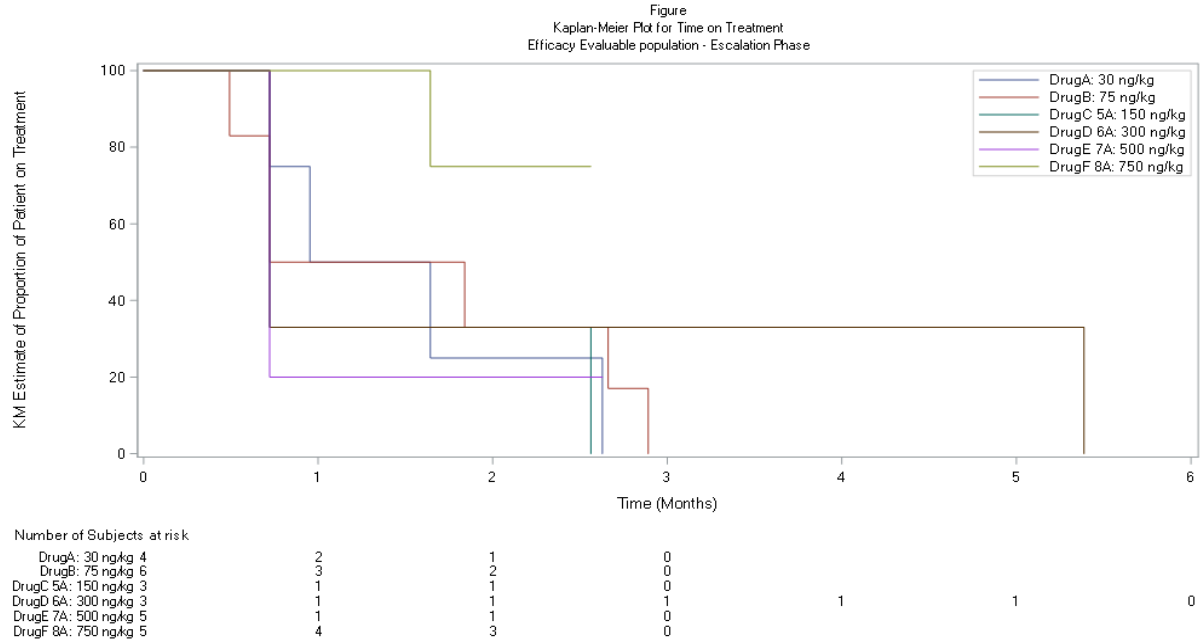


PLOT Example #3: Common legend zoom

```
proc template;
define statgraph temp;
begingraph;
  layout lattice / rows = 2 rowdatarange=data columndatarange=data rowgutter=4px columngutter=2px
                   rowweights=(75 30 100);

/** TOTAL BILIRUBIN PLOT ***/
    layout overlay / xaxisopts=(display=none) yaxisopts=( griddisplay=on label=('BILIRUBIN (umol/L)'))
                     walldisplay=(FILL);
      Series plot of bilirubin……..;
      scatterplot of bilirubin……..;
          /* if you want a plot with lines and markers you can use just the Series statement with the Markers option
          and marker attribute statements added. Scatter plot can be avoided this way, unless you need upper/lower
          error bars.*/
 endlayout;

/** COMMON LEGEND PLOT ***/
            layout overlay;
              discretelegend " var"  /
                                  border = true displayclipped = true across = 4 halign = right;
            endlayout;

/** ALT/AST PLOT ***/
 layout overlay / xaxisopts=(display=(line ticks tickvalues label) label=('Time (Cycle/Day)'))
     yaxisopts=( griddisplay=on label=('AST/ALT (IU/ML)')) walldisplay=(FILL);
           Seris plot of ALT/AST……;
           scatterplot ALT/AST……..;
    endlayout;
  endlayout;
endgraph;
end; run;
```

Example #4: Displaying ATRISK data in a survival plot without using annotation facility.

Kaplan-Meier plots tend to me more informative when At-Risk subject data is displayed on the timeline. Traditionally this was done using the annotation facility in SAS/Graph. However, the use of BLOCKPLOT feature to display At-Risk data can make it very easy to combine the K-M Plot and At-Risk data in GTL. When it is meaningful to display a zero for missing data in a time series the **repeatedvalues=true** option is specified in the BLOCKPLOT statement.



PLOT Example #4: Displaying ATRISK data in a survival plot without using annotation facility.

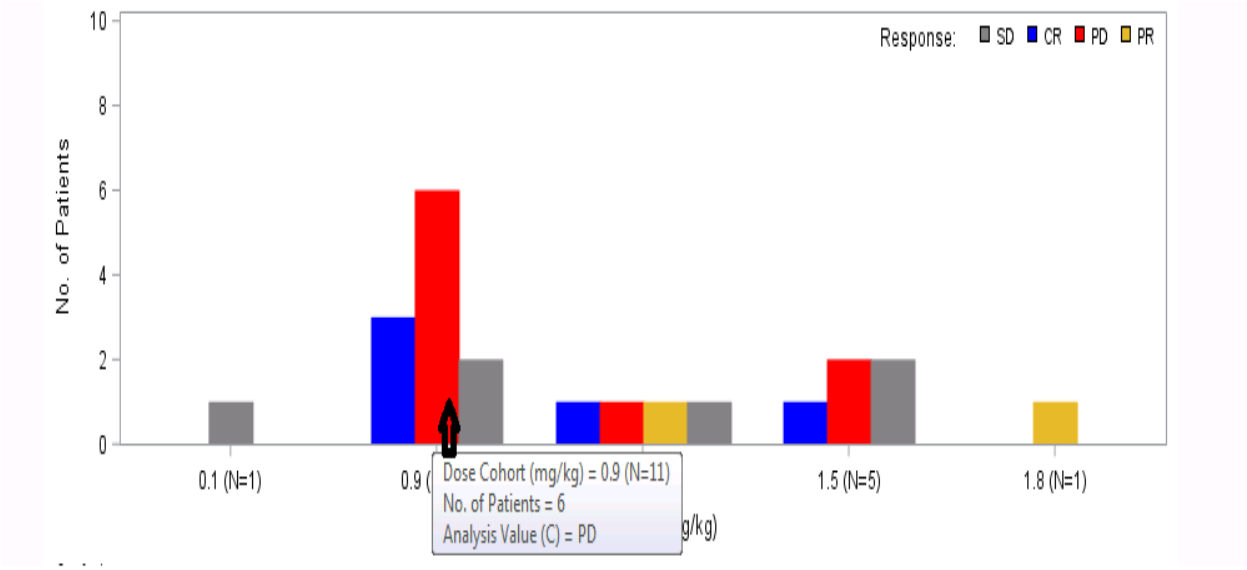
```
proc template;
define statgraph xendesign;
begingraph;

layout lattice / rowdatarange=data columndatarange=union rowgutter=10 columngutter=10
                 rowweights=(0.80 0.20) rows=2;
  layout overlay / xaxisopts=(label=('Time (Months)') linearopts=(tickvaluelist=(0 1 2 3 4 5 6 7
                              8 9 10 11 12 13 14)))
   yaxisopts=( label=("&yaxis") linearopts=(tickvaluesequence=(start=0 end=100 increment=20)));
   stepplot x=time y=survival /  group = StratumNum name='step' connectorder=xaxis ;
  discretelegend 'step'  'scatter'/ opaque=false border=true halign=right valign=top displayclipped=true
  across=1 order=rowmajor location=inside ;
 endlayout;

  layout overlay / walldisplay=none xaxisopts=(display=none) ;
       entry halign=left "Number of Subjects at risk" / location=outside valign=top;
       blockplot x=tatrisk block=atrisk / display=(label values)
                 class=stratumnum repeatedvalues=true valuehalign=start valueattrs=(size=8)
              labelattrs=(size=8);
   endlayout;

endlayout;
endgraph;
end;
run;
```

Add Tooltips to Graphs for display in HTML:

Often statisticians and CDM managers have questions about the underlying data that is displayed in a graph. Informative descriptions can be inserted that can be made visible when a mouse is hovered over certain areas of the graph. When you specify the HTML destination and IMAGEMAP=ON in the ODS GRAPHICS statement, an image map of coordinates for tooltips is generated along with the HTML output file.

```
ods html body=' C:\Figures\fig.html' style=HTMLBlue;
ods graphics on / imagemap=on;
        …
        … SAS Graph Statements
        …
ods html close;
```

DISPLAY 5: Demographics bar chart with tooltips



DISPLAY 6: Grouped bar chart with tooltips showing information for each bar.

Combine Multiple Graphs into a single ODS output file:

By using the ODS Document feature we can combine individual plots from multiple locations or procedures into a single document. Enclose your SAS Graph production code with an ODS DOCUMENT statement for each figure.

```
ods document name=test.KM1;
        …..
        Graph procedures/statements;
        …..
ods document close;
```

Once the plots are produced by the statistics procedures, the next step is to combine these plots in the target file.

Enclose PROC DOCUMENT calls within the ODS PDF or ODS RTF target file statements. Use PROC DOCUMENT to use the plots available by calling them with the same name used by the ODS DOCUMENT statement.

```
ods pdf file="C:\Figures\Figures.pdf";

/*assemble individual graphs using ods document and close; */
proc document name=test.KM1;
  replay;
run;
quit;

proc document name=test.KM2;
  replay;
run;
quit;
ods pdf close;
```

CONCLUSION

ODS Graphics and Statistical procedures can produce graphical output together with tabular output. The examples above illustrate the power of ODS Graphics to create complex graphs without the need to learn how to write complex SAS/Graph code.

REFERENCES

SAS/STAT® 14.1 User's Guide Statistical Graphics Using ODS
Basic ODS Graphics Examples Warren F. Kuhfeld
Advanced ODS Graphics Examples Warren F. Kuhfeld
Kuhfeld, W. F. (2010). Statistical Graphics in SAS: An Introduction to the Graph Template Language and the Statistical Graphics Procedures. Cary, NC: SAS Institute Inc.
Matange, S., and Heath, D. (2011). Statistical Graphics Procedures by Example: Effective Graphs Using SAS. Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

I would like to thank Robert Diseker, Ajay Sadey and Ed Foundos for their inputs and review.

CONTACT INFORMATION

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

Harivardhan Jampala
Chiltern, a Covance Company
4000 CentreGreen Way, Suite 300, Cary, North Carolina, 27513,
United States: +1 919 462 8867
Email: Hari.Vardhan@Chiltern.com