An efficient way to create graphs in SAS 9.2: Utilizing SG procedures and GTL

Yunzhi Ling, Sanofi-aventis, Bridgewater, NJ

ABSTRACT

In SAS 9.2, the new SAS/GRAPH SG procedures and the Graph Template Language (GTL) become production. The SG procedures, which are built on GTL, support a clear and concise syntax for creating many types of graphs. While the SG procedures are powerful, they do not cover all options in GTL and may not meet users' specific requirements. For a graph that is beyond the scope of the SG procedures, a customized graph template written in GTL syntax is a must. Through examples, this paper shows how to utilize the SG procedures and GTL simultaneously to create some common statistical graphs in clinical research.

INTRODUCTION

With SAS 9.2, many graphs can be quickly produced using the new SAS/GRAPH SG procedures and the Graph Template Language (GTL). GTL is a very powerful language to create sophisticated analytical graphs and is the foundation of many SAS procedures to produce graphical output, including the new SG procedures. The SG procedures are designed to create stand-alone graphs that complement the more specialized graphs produced by the statistical procedures that use the ODS Statistical Graphics infrastructure. The SG procedures, including SGPLOT, SGPANEL, SGSCATTER and SGRENDER, enable users to easily create many types of graphs as following:

- The SGPLOT procedure creates single-cell plots and charts with overlay capabilities, e.g., scatter, series, step, band, needle, box blot, histogram, dot plots, bar charts, normal curve, loess, regression, etc.
- The SGPANEL procedure creates paneled graphs driven by class variables; the plots contained within each panel are similar to the plots from the SGPLOT procedure.
- The SGSCATTER procedure creates paneled scatter plots, with overlay fits and confidences.
- The SGRENDER procedure creates customized plots by associating a user-defined template written in GTL with a dataset.

For a graph that can not be obtained directly from the SG procedures, a user-defined graph template based on GTL is needed. However, GTL encompasses a large amount of statement and options, and is a relatively new language to many SAS users. To avoid writing GTL codes from scratch, one practical approach is to combine the SG procedures with GTL for a desired graph, in these steps:

1. Use SG procedure to sketch out the plot, output the underlying GTL syntax into a graph template;
2. Customize the graph template to the desired layout;
3. Use Proc SGRENDER to associate the template with a dataset for graph creation.

This blended approach shortens the GTL learning curve and gives flexibility and simplicity to create customized graphs.

GTL BASICS

In the Graph Template Language, the STATGRAPH template, a reserved graph template from the TEMPLATE procedure, can be defined, compiled, and associated with a dataset for graph creation. The STATGRAPH template describes the layout and appearance of a graph, using a structured building-block approach. Two types of graph, single-cell and multi-cell, can be defined in the STATGRAPH template, with a cell representing a distinct sub-region of a graph that contains plots, text and legends.

A single-cell graph has only one block taking the whole graph region. The OVERLAY layout is the most essential single-cell layout. Other single-cell layouts include OVERLAY3D, OVERLAYEQUATED, and PROTOTYPE. The graphs created by the SGPLOT procedure are single-cell graphs.

On the other hand, a multi-cell graph has multiple cells sharing one graph region. Among four types of multi-cell layouts (LATTICE, DATALATTICE, GRIDDED and DATAPANEL), the LATTICE layout is the most frequently used layout. The SGPANEL procedure and the SGSCATTER procedure generate multi-cell graphs.

The graph templates of this paper focus on two commonly used graphical layouts: the OVERLAY layout for single-cell graph and the LATTICE layout for multi-cell graph.

Basic syntax of the STATGRAPH template for the OVERLAY layout:

```
*** step 1. define graph template;
proc template;
  define statgraph name-of-graph-template;
  entrytitle statement;
  begingraph;
    layout overlay / <options>;
        Plot statements; - including plots and legends statement;
    endlayout;
    entryfootnote statement;
  endgraph;
end;

*** step 2. connect the template with a dataset for graph creation;
proc sgrender data=input-dataset template=name-of-graph-template;
run;
```

Basic syntax of the LATTICE layout graph template:

```
*** step 1. define graph template;
proc template;
  define statgraph name-of-graph-template;
  mvar name(s); - define macro variables;
  dynamic name(s); - define dynamics;
  entrytitle statement;
  begingraph;
    layout lattice/ <options>;
       layout overlay;
           plot statements;
       endlayout;
       more overlay layouts and/or plot statements;
    endlayout;
    entryfootnote statement;
  endgraph;
end;

*** step 2. connect the template with a dataset for graph creation;
proc sgrender data=input-dataset template=name-of-graph-template;
run;
```

The following examples demonstrate how to create graphs either directly from SG procedures or through some user-defined graph templates.

EXAMPLE 1: USE SG PROCEDURES TO CREATE BAR CHARTS OF ADVERSE EVENTS

Bar charts can be easily created in SAS with Proc SGPLOT or Proc SGPANEL.

Proc SGPLOT in Program 1.1 produces bar charts of number of patients who experienced an adverse event in body system organ class, as shown in Figure 1.1. The HBAR creates horizontal bar charts of TEAE (response) by AEBODSYS (category), with the response values summed by category. The bars are divided into two treatment groups because of the GROUP=TRT option. Given no destination is specified, the output graph goes to the default LISTING destination.

Program 1.2 uses Proc SGPANEL to create similar bar charts as shown in Figure 1.2. With PANELBY TRT statement, the bars are displayed separately by treatment group for easy comparison. The graph is sent to RTF destination using ODS RTF statement.

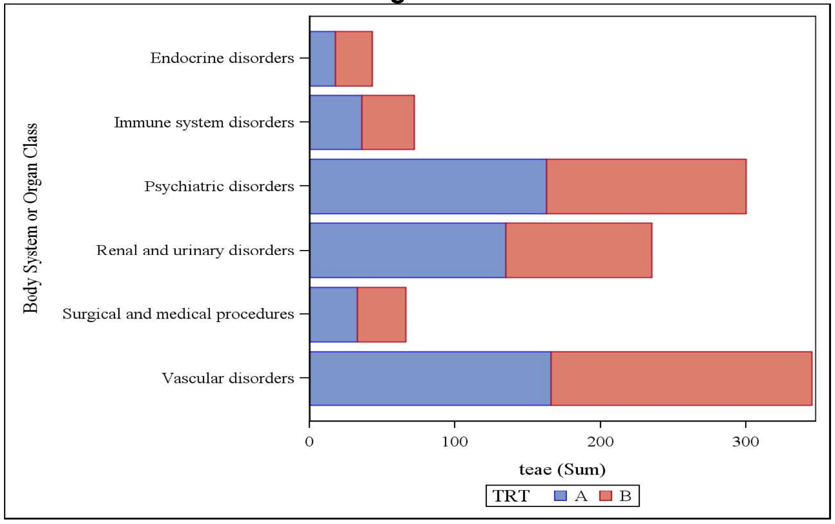
The input data set 'ADAE' has the following key variables:

```
SUBID   TRTN TRT  AETERM                     AEBODSYS                   TEAE
 001      1   A    WORSENING HYPERTENSION     Vascular disorders           0
 001      1   A    DEPRESSION                 Psychiatric disorders        1
 002      2   B    NEPHRITIS                  Renal and urinary disorders  1
 002      2   B    HYPERTENSION               Vascular disorders           1
 ...
```

Program 1.1

```
proc sgplot data=adae;
  hbar aebodsys /group=trt
  response=teae stat=sum;
run;
```

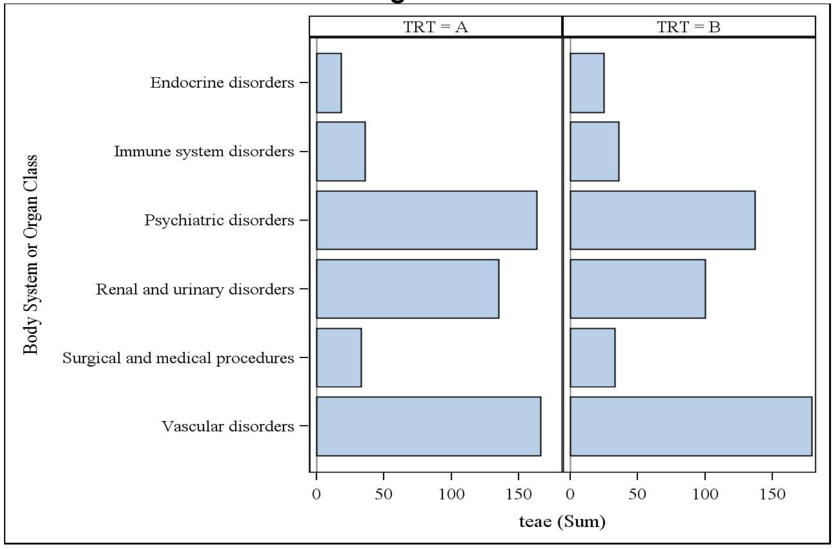
Figure 1.1



Program 1.2

```
ods rtf file="C:\aebar2.rtf";
proc sgpanel data=adae;
  panelby trt;
  hbar aebodsys /
  response=teae stat=sum;
run;
ods rtf close;
```

Figure 1.2



EXAMPLE 2: COMBINE SGPLOT AND GTL SYNTAX TO CREATE SINGLE-CELL PLOT – LINE PLOT OF PK/PD PROFILE

This example demonstrates how to combine SGPLOT and GTL syntax to create a Pharmacokinetic (PK) / Pharmacodynamic (PD) profile, an overlaid line plot of PK concentration and ∆QTcF (change from baseline in QTcF) over a time interval. In Program 2.1, the data step prepares the data: 1) shift TIME value for one treatment group to separate two error bars at each time point; 2) calculate variables QTCLOW and QTCHIGH for error bars since Proc SGPLOT has no calculation function. Proc SGPLOT produces two series of line plot overlaid on one horizontal axis with two reference lines based on the Y axis, as shown in Figure 2.1.

- The TMPLOUT=FILENAME option sends the GTL codes of the Proc SGPLOT into temp_pkpd.sas.
- The XAXIS, YAXIS and Y2AXIS statements specify the axis tick values and label.
- Special symbols ∆ and ± are inserted in YAXIS label and TITLE through (*ESC*) {UNICODE}. Prefix (*ESC*) to {UNICODE} is the only way to insert symbols in the SG procedures.
- The REFLINE statement adds two reference lines at Y=0 and Y=10 of YAXIS, respectively.
- The mean +/- error bar plot for variable QTC against YAXIS is produced by overlaying a SERIES and SCATTER plot because the SERIES statement does not support error bars. The first SERIES statement creates a line plot connecting QTC values. The SCATTER statement displays the error bars of QTC values. The plot's attributes, i.e., colors, marker symbols and line patterns, can not be user-specified through MARKERS and LINEATTRS= options due to GROUP=TRT usage. They are automatically assigned from graphical style elements GraphData1 and GraphData2 in the underlying GTL syntax.
- The second SERIES statement creates a line plot of CONC value based on Y2AXIS.
- The KEYLEGEND statements add legends to the plot. The name argument corresponds to the unique name that is assigned to each plot.

The input data set 'PKPD' contains these variables:

```
TRTN  TRT     TIME   QTC   SEM   CONC
  1   Trt A     1     2    0.5    100
  2   Trt B     1    -1    0.8    120
  1   Trt A     2     6      1    320
  2   Trt B     2     3    0.9    160
 …
```

Program 2.1

```
data pkpd;
     set pkpd;
     if trtn=1 then time=time+0.2;
    qtclow=qtc-sem;  qtchigh=qtc+sem;
run;
proc sgplot data=pkpd tmplout="C:\temp_pkpd.sas";
     xaxis  values=(0 to 9 by 1, 12, 13) label="Time (hours post dose)" ;
     yaxis  values=(-14 to 14 by 2) label="(*ESC*){unicode delta_u}QTcF TM (ms,
             mean(*ESC*){unicode '00B1'x}sem)";
     y2axis values=(0 to 500 by 100) label="Conc. (ng/mL, mean)" ;
    refline 0 10 /axis=y;
     series x=time y=qtc / group=trt name="series";
     scatter x=time y=qtc/roup=trt yerrorlower=qtclow yerrorupper=qtchigh
                          name="scatter";
     series x=time y=conc /group=trt y2axis name="series1" ;
     Title "Time profiles: (*ESC*){unicode delta_u}QTcF mean(*ESC*){unicode
            '00B1'x}sem, conc. mean ";
    keylegend "series" "scatter"/ position=bottom;
    keylegend "series1"/ position=bottom;
 run;
```

Figure 2.1

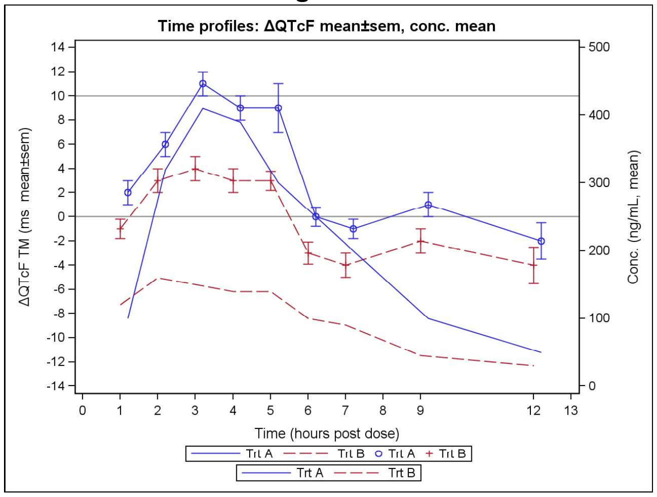


Figure 2.2

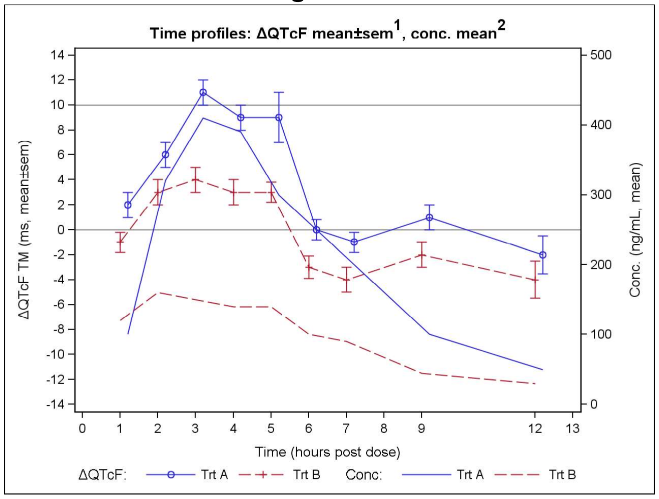


Figure 2.1 is very close to the target graph displayed in Figure 2.2, except for the display of consolidated legends to differentiate the lines, Δ symbol in the legend and two superscripts in the title. This is achieved by modifying the template temp_pkpd.sas obtained from Program 2.1 into a user-defined template called PKPDPLOT as presented in Program 2.2. The key changes are:

- Insert text commands {SUP '1'} and {SUP '2'}, and remove the redundant prefix (*ESC*) of hexadecimal Unicode in ENTRYTITLE. In fact, special text commands such as {SUP}, {SUB} and {UNICODE} only work directly in ENTRY, ENTRYTITLE and ENTRYFOOTNOTE text statement. To use Unicode characters in Labels, as shown in YAXIS label and legend title, an in-line formatting strategy is required. This so-called in-line formatting means embedding an ODS escape sequence (*ESC*) before {UNICODE} in a quoted string, which signals that the next token represents a text command. Currently, only the {UNICODE} text command is recognized by (*ESC*), not {SUP} or {SUB}.
- Use EVAL function to calculate YERRORUPPER and YERRORLOWER for the upper and lower error bars. Supporting expressions and functions in GTL exerts less restriction on input data.
- In DISCRETELEGEND statements, MERGE=TRUE combines the marker and line entries, TITLE= adds title to legend, HALIGN= sets alignment for two legends so they can fit into one line, and BORDER=NO option hides the border.
- After template PKPDPLOT is defined, use PROC SGRENDER to associate the template with input dataset PKPD for Figure 2.2 creation.

Program 2.2

```
proc template;
define statgraph pkpdplot;
begingraph;
   EntryTitle 'Time profiles: '{unicode delta_u}'QTcF mean'{unicode '00B1'x}'
              sem'{sup '1'}', conc. mean'{sup '2'}';
   layout overlay /xaxisopts=(label="Time (hours post dose)" type=linear
                              linearopts=(tickvaluelist=(0 1 2 3 4 5 6 7 8 9 12 13)
                              viewmin=0 viewmax=13 ))
                  yaxisopts=(label="(*ESC*){unicode delta_u}QTcF TM(ms,
                              mean(*ESC*){unicode '00B1'x}sem)"  type=linear
                              linearopts=(tickvaluelist=( -14 -12 -10 -8 -6 -4 -
                              2 0 2 4 6 8 10 12 14 ) viewmin=-14 viewmax=14 ))
                  y2axisopts=(label="Conc. (ng/mL, mean)" type=linear
                             linearopts=( tickvaluelist=( 0 100 200 300 400 500 )
                             viewmin=0 viewmax=500 ));
     ReferenceLine y=0 / clip=true;
     ReferenceLine y=10 / clip=true;
```

```
        SeriesPlot X=time Y=qtc / Group=trt LegendLabel="qtc" NAME="SERIES";
        ScatterPlot X=time Y=qtc /primary=true Group=trt YErrorUpper=eval(qtc+sem)
                        YErrorLower=eval(qtc-sem)   LegendLabel="qtc" NAME="SCATTER";
        SeriesPlot X=time Y=conc / yaxis=y2 Group=trt LegendLabel="conc"
                                   NAME="SERIES1";
        DiscreteLegend "SERIES" "SCATTER" / merge=true title="(*ESC*){unicode
              delta_u}QTcF:" Location=Outside valign=bottom halign=left border=no;
        DiscreteLegend "SERIES1" / Location=Outside valign=bottom halign=right
                                  title="Conc:" border=no;
     endlayout;
   endgraph;
   end; run;

   *** Submit PROC SGRENDER to create graph;
   ods rtf file="C:\pkpdplot.rtf";
   proc sgrender data=pkpd template=pkpdplot;
   run;
   ods rtf close;
```

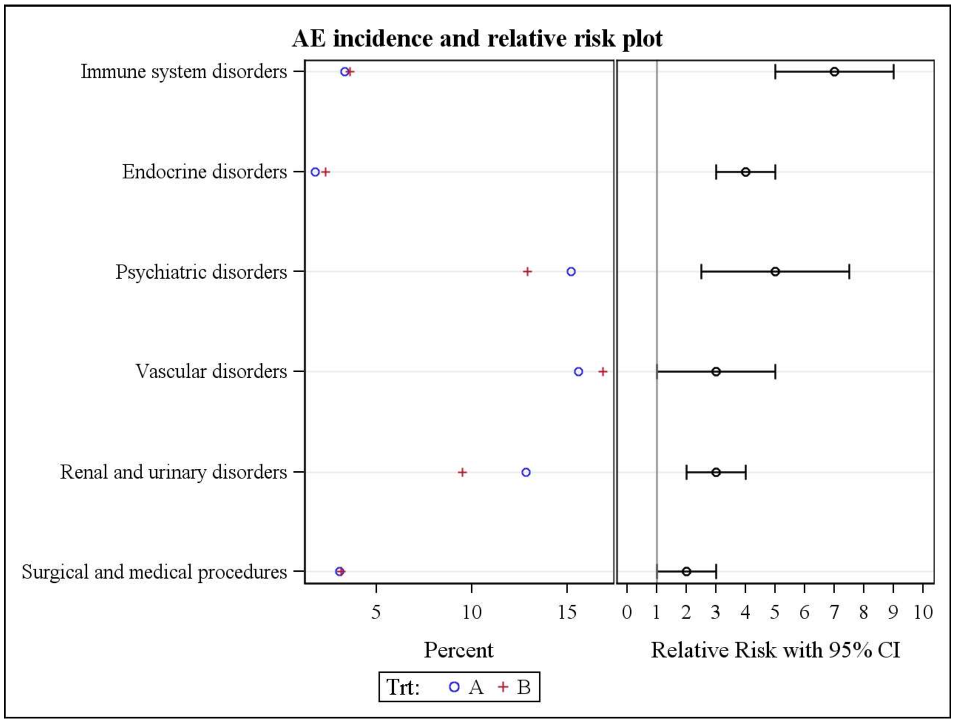
EXAMPLE 3: COMBINE SGPLOT AND GTL SYNTAX TO CREATE MULTI-CELL GRAPH – AE INCIDENCE AND RELATIVE RISK PLOT

Figure 3 displays a summary plot of AE incidence and relative risk, which is a typical multi-cell graph, with a dot plot of AE incidence and a scatter plot of relative risk displayed side-by-side. This example shows how to put two single plots horizontally into one graph.

The input data set 'ADAE2' contains these variables:

```
TRT AEORDER AEBODSYS                  COUNT  PERCENT  RRISK  RRISKSD  RRLOW  RRHIGH
 A     4    Immune system disorders     36     3.32     7        2      5       9
 B     4    Immune system disorders     39     3.59     7        2      5       9
 A     5    Endocrine disorders         19     1.75     4        1      3       5
 B     5    Endocrine disorders         25     2.30     4      1.5    2.5     5.5
...
```

Figure 3



The SG procedures in Program 3.1 and 3.2 are very straightforward, with the corresponding GTL codes saved into ae_pct.sas and ae_rr.sas. It is worthy mentioning that the sorting order of input dataset based on variable AEORDER controls the AE body system code appearance along the vertical axis.

Program 3.1

```
Proc sort data=adae2;
  by aeorder;
run;

proc sgplot data=adae2
     tmplout="C:\ae_pct.sas";
  scatter x=percent
         y=aebodsys/group=trt;
 yaxis discreteorder=data
     display=(nolabel) grid;
  xaxis label='Percent';
run;
```

Figure 3.1

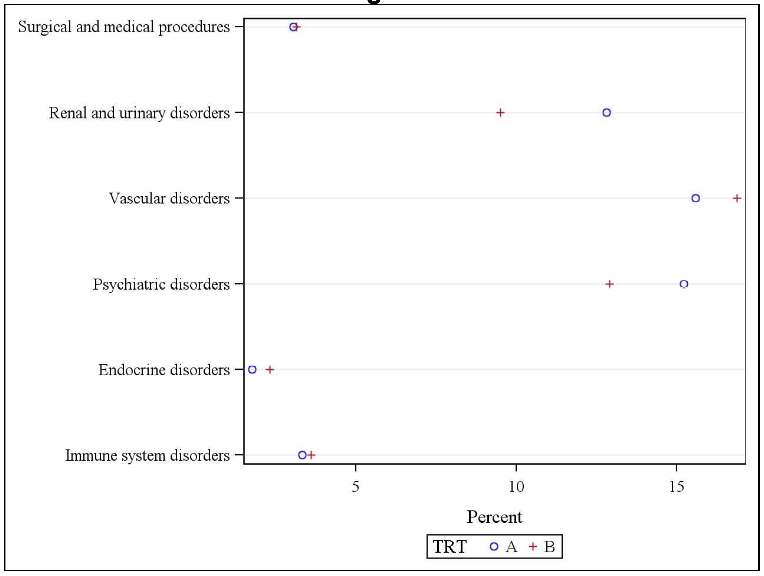
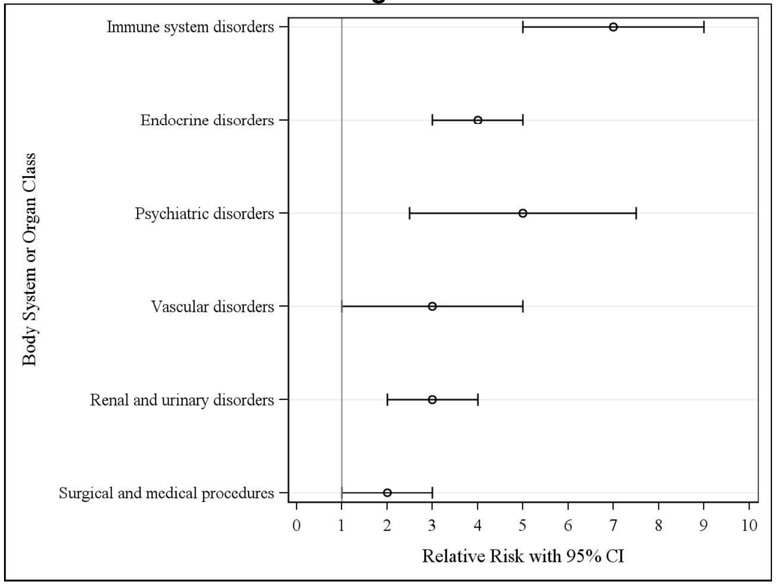


Figure 3.2



Program 3.2

```
Proc sort data=adae2;
  by descending aeorder;
run;

proc sgplot data=adae2
     tmplout="C:\ae_rr.sas";
  scatter x=rrisk y=aebodsys
          /xerrorlower=rrlow
          xerrorupper=rrhigh;
  xaxis values=(0 to 10 by 1)
        label='Relative Risk
        with 95% CI';
  yaxis grid;
  refline 1 /axis=x;
run;
```

To define a template for Figure 3, the overlay layout statements from two single-cell graphs, ae_pct.sas and ae_rr.sas, are extracted and then inserted into a two-column-one-row lattice layout template with key features highlighted in Program 3.3.

- Set the lattice layout as two columns and one row, with two columns apportioned at 65% and 35% to balance the two plots and to properly display the body system organ class along the first YAXIS.
- Change the name of the first SCATTERPLOT statement from SCATTER to SCATTER1 to differentiate from the name of the second SCATTERPLOT statement in order to properly assign legend to AE incidence plot.
- DISPLAY=NONE for the second YAXIS options conceals the YAXIS for the relative risk plot.

Program 3.3

```
proc template;
define statgraph aeplot;
 begingraph;
   entrytitle "AE incidence and relative risk plot";
   layout lattice / columns=2 rows=1 columnweights=(.65 .35);
       *** overlay layout codes obtained from ae_pct.sas;
       layout overlay / xaxisopts=(Label="Percent" type=linear)
                        yaxisopts=(griddisplay=on display=(ticks tickvalues line)
                        type=discrete);
          ScatterPlot X=PERCENT Y=AEBODSYS / primary=true Group=TRT
                                                NAME="SCATTER1";
          DiscreteLegend "SCATTER1"/ Location=outside valign=bottom title="Trt:";
       endlayout;
       *** overlay layout codes obtained from ae_rr.sas;
       layout overlay / xaxisopts=( Label="Relative Risk with 95% CI" type=linear
                        linearopts=( tickvaluelist=( 0 1 2 3 4 5 6 7 8 9 10 )
                        viewmin=0 viewmax=10 ))
                        yaxisopts=( griddisplay=on display=none);
          ScatterPlot X=rrisk Y=AEBODSYS / primary=true XErrorUpper=rrhigh
                                    XErrorLower=rrlow  NAME="SCATTER";
          referenceLine x=1/ clip=true;
       endlayout;
    endlayout; **lattice;
 endgraph;
end;
run;

*** sort the input dataset and submit PROC SGRENDER to create the graph;
ods rtf file="C:/aeplot.rtf";
proc sort data=adae2;
  by descending aeorder;
run;
proc sgrender data=adae2 template=aeplot;
run;
ods rtf close;
```

EXAMPLE 4: COMBINE SGPLOT AND GTL TO CREATE MULTI-CELL GRAPH - MEAN PLOT OVER TIME WITH EXPOSED PATIENTS DISPLAYED AT EACH TIMEPOINT

Mean plot over time with the number of exposed patients displayed at each time point is widely used in clinical studies. This example explains how to create mean plot of two vital signs parameters, weight and heart rate, separately. Other plots of similar pattern, e.g., exposure plot and survival plot, can be generated using the same approach.

The input data set 'ADVS' contains these key variables:

```
TRTGRP  VSTESTCD  VSTEST      VISIT    MEAN   STDERR   COUNT
Trt A          1  Weight          2      50        2     177
Trt B          1  Weight          2      55        1     174
...
Trt A          2  HeartRate       6      80        2      88
Trt B          2  HeartRate       6      90        3     101
...
```

In Program 4.1, the data step before Proc SGPLOT prepares the data: 1) calculate YLOW and YHIGH for error bars of the scatter plot; 2) create variable VISIT2 to separate two error bars for two treatment groups, and variable VISIT3 equals to variable VISIT except it is set to missing for visit 7 to disconnect lines between visit 6 and visit 7. Two separate graphs are generated: the first Proc SGPLOT, similar to Program 2.1, produces mean plot with standard error bars over time as shown in Figure 4.1; and the second Proc SGPLOT creates a scatter plot of the patient counts as Figure 4.2, with the following specialties:

- The DATALABEL=count option displays the counts of each TRTGRP as the label for each visit.
- Specify COLOR=WHITE in MARKERATTRS= option to make the marker of scatter plot invisible; otherwise, a circle would show up next to each count.
- The tick values of XAXIS take visit format in display.

Program 4.1

```
proc format;
     value visfmt
       2='Baseline'  3='V2' ...  7='LOCF'  other=" " ;
data advs;
     set advs;
     ylow=mean-stderr;  yhigh=mean+stderr;
    if substr(trtgrp, 5, 1)='A' then visit2=visit;
     if substr(trtgrp, 5, 1)='B' then visit2=visit+0.1;
    if visit ne 7 then visit3=visit;
run;

*** top part, show in Figure 4.1;
proc sgplot data=advs(where=(vstestcd=1)) tmplout="C:\temp_top.sas";
     format visit2 visfmt.;
     xaxis values=(1 to 8 by 1) label="Days in the study";
     series x=visit3 y=mean / group=trtgrp;
     scatter x=visit2 y=mean / group=trtgrp yerrorlower=ylow yerrorupper=yhigh;
     keylegend / position=bottom title="Treatment group:";
run;
*** bottom part, show in Figure 4.2;
proc sgplot data=advs(where=(vstestcd=1)) tmplout="C:\temp_bottom.sas";
     format visit visfmt.;
     xaxis values=(1 to 8 by 1) ;
     scatter x=visit y=trtgrp/ datalabel=count markerattrs=(color=white);
run;
```

Figure 4.1

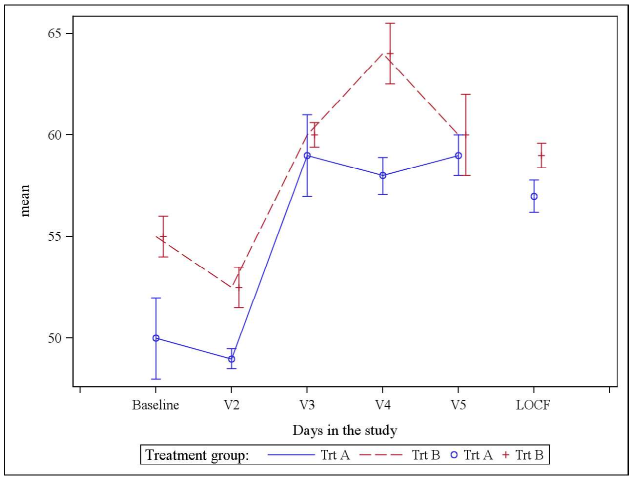
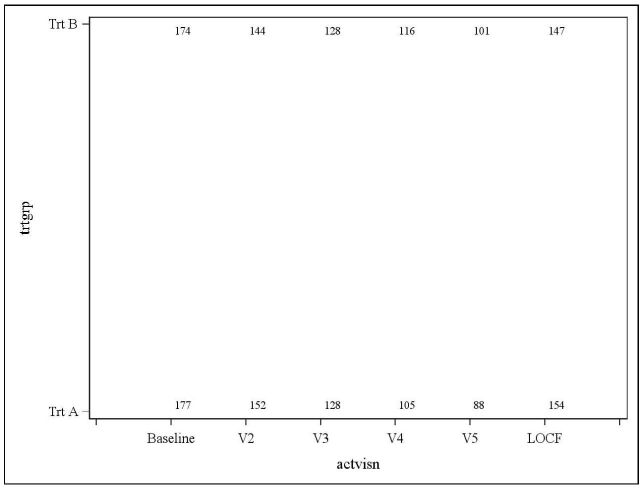


Figure 4.2



Next, define template MEANPLOT to stack Figure 4.1 and Figure 4.2 vertically into one graph. Following the same strategy in the previous example, the overlay layout statements of Figure 4.1 and 4.2 are extracted and placed into a one-column-two-row lattice layout template as shown in Program 4.2. Some important features of this template are:

- Set the lattice layout as one column and two rows with a common horizontal axis. Two rows are split proportionally at 85% and 15% to properly display the top and bottom part of the graph.
- MVAR and DYNAMIC statement define macro variables and dynamics to make the template more flexible and data independent. The macro variables and dynamics are resolved when the template is executed.
- WALLDISPLAY=NONE and BORDER=FALSE eliminate the edge of the patient counts plot.
- DISPLAY= (TICKVALUES) makes only the YAXIS tick values, TRTA and TRTB, appear in the patient counts plot.
- After the template MEANPLOT is defined, put Proc SGRENDER into macro %plot so it can be associated with multiple parameters.
- DYNAMIC statement of Proc SGRENDER supplies value for dynamics. (*ESC*){UNICODE} displays ± symbol in dynamics _YLABEL.
- Call macro %plot on weight and heart rate separately to generate Figures 4.3 and 4.4.

Program 4.2

```
proc template;
define statgraph meanplot;
 mvar sysdate9;
 dynamic _x1 _x2 _x3 _y _ylabel _count _title _footnote;
 begingraph;
   entrytitle _title;
   layout lattice / columns=1 rows=2 rowweights=(.85 .15);
       **** code obtained from temp_top.sas;
       layout overlay / xaxisopts=( Label="Days in the study" type=linear
                                    linearopts=( tickvaluelist=( 1 2 3 4 5 6 7 8 )
                                    viewmin=1 viewmax=8 ) )
                       yaxisopts=( Label=_ylabel type=linear );
           SeriesPlot X=_x3 Y=_y / Group=trtgrp Markerattrs=( Symbol=CIRCLE)
                                   LegendLabel="mean" NAME="SERIES";
           ScatterPlot X=_x2 Y=_y / primary=true Group=trtgrp
                                   YErrorUpper=eval(mean+stderr)
                                   YErrorLower=eval(mean-stderr)
                                   LegendLabel="mean" NAME="SCATTER";
           DiscreteLegend "SERIES" "SCATTER" / merge=true border=no
                          Location=outside valign=bottom title="Treatment Group:";
```

```
            endlayout;
            **** code obtained from temp_bottom.sas;
            layout overlay / walldisplay=none border=false
                             xaxisopts=( display=none type=linear
                                         linearopts=( tickvaluelist=( 1 2 3 4 5 6 7 8 )
                                         viewmin=1 viewmax=8 ) )
                             yaxisopts=( display=(tickvalues) type=discrete );
                scatterplot X=_x1 Y=trtgrp/ DataLabel=_count primary=true
                                         Markerattrs=( Color=WHITE) DataTransparency=0;
            endlayout;
        endlayout; ****end of lattice;
        entryfootnote halign=left _footnote " (" sysdate9 ")";
  endgraph;
end;
run;

**** put PROC SGRENGER into a macro to associate with multiple parameters;
%let pgm=meanplot.sas;
%macro plot(vstest=, vstestcd=);
ods rtf file="C:/meanplot&vstest..rtf";
proc sgrender data=advs(where=(vstestcd=&vstestcd.)) template=meanplot;
   format visit visit2 visit3 visfmt.;
   dynamic _x1="visit" _x2="visit2" _x3="visit3" _y="mean"
           _ylabel="&vstest. (mean(*ESC*){unicode '00B1'x}sem)" _count="count"
           _title="Change from baseline in &vstest."
           _footnote="program=&pgm., output=meanplot&vstest..rtf";
run;
ods rtf close;
%mend;
**** run macro %plot on weight and heart rate to create figure 4.3 and 4.4;
%plot(vstest=Weight, vstestcd=1)
%plot(vstest=HeartRate, vstestcd=2)
```

Figure 4.3

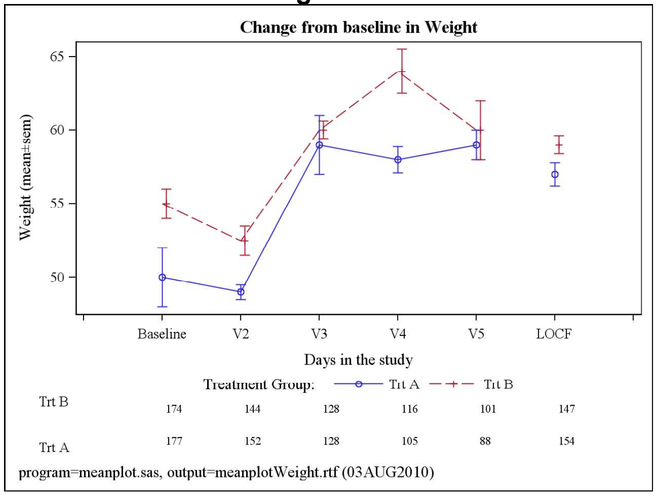
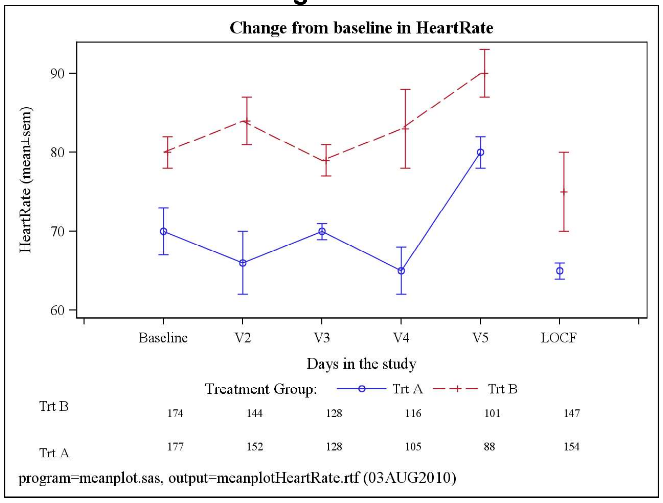


Figure 4.4



CONCLUSIONS

The SG procedures and the Graph Template Language provide an efficient way of creating graphs compared with traditional SAS/GRAPH procedures. Adapting a graph template from SG procedures into a user-defined template provides a fast yet flexible approach to create customized graphs. This paper covered a few examples of statistical graphs commonly used in clinical research. The programs can be tailored for other statistical graphs by following the same principle.

REFERENCES

Dan Heath, "Secrets of the SG Procedures", Paper presented at SAS Global Forum 2009

Jeff Cartier, "A programmer's Introduction to the Graphics Template Language", Paper presented at SUGI 31

Sanjay Matange, "Introduction to the Graph Template Language", Paper presented at SAS Global Forum 2008

ACKNOWLEDGMENTS

I am grateful to Cindy Song of Sanofi-aventis for her thorough review and invaluable comments.

CONTACT INFORMATION

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

Yunzhi Ling
Sanofi-aventis
BX2-416B, 200 Crossing Boulevard
Bridgewater, NJ 08807-0890
908-304-6488
Yunzhi.ling@sanofi-aventis.come