Paper DV02

Graphs made easy using SAS Graph Template Language

Manohar Modem, Cytel Inc;
Bhavana Bommisetty, Vita Data Sciences

ABSTRACT

In clinical domain, we used to create graphs using traditional SAS/GRAPH procedures like PROC GPLOT. To create better graphs, SAS developed Graph Template Language (GTL). GTL has many significant advantages over traditional SAS/GRAPH procedures like more control over visual attributes of the graph, less annotation and minimal coding. Statistical Graphics (SG) procedures like PROC SGPLOT, SGPANEL etc., which require minimal coding use GTL to create commonly used graphs. But, writing graphs in GTL has more advantages compared to SG procedures and traditional SAS/GRAPH procedures. GTL code might look a little lengthy and intimidating, but once you understand how to read GTL code, it will be very useful to create not only simple graphs but also complex graphs. The purpose of this paper is to help the programmer understand and utilize various elements of GTL to create wonderful graphs.

INTRODUCTION

Creating SAS graphs needs us to look at various elements involved in a graph. It is quite simple to find descriptive statistics using SAS procedures like proc freq/proc means as the number of SAS statements and options required is very few and can be easily remembered. But SAS graph procedures has many statements and many more options. So, it isn't easy to remember all these statements and options and when to use them correctly. Therefore, we need to a have a specific approach on how to create SAS graphs. This paper demonstrates how to do various clinical graphs using GTL. Besides providing GTL code and graphs, we have also provided dummy data that was used to create these graphs. The idea behind providing the dummy data is to encourage the programmer to generate these graphs by themselves and play with the code and see how various statements and options work. SAS code, dummy input data and the figure outputs are stored in GitHub repository:

https://github.com/mmodem/SAS-graphs-using-GTL.git

In creating common clinical graphs using GTL, this paper covers the following topics:

1. Different types of Annotation.
2. Methods to create/modify attributes of GTL elements.
3. Usage of dynamic variables
4. Conditional statements in proc template
5. Single and Multicell (custom/panel) graphs.
6. Usage of appropriate options for various elements

We use two procedures to create graphs in GTL.

1. Proc template
2. Proc sgrender

By itself, proc template doesn't create the graph. It creates a template and we apply this template to the data using proc sgrender.

Here is the basic code that is needed in writing GTL code:

```
proc template;
  define statgraph template-name;
  begingraph;
    …SAS statements…
  endgraph;
  end;
run;

proc sgrender data= template= template-name;
run;
```

The common elements and their corresponding proc template statements in a SAS graph are:

	Element	GTL statement
1	Title	Entrytitle
2	Footnote	Entryfoonote
3	Plot	Depends on the graph. E.g. scatterplot
4	Legend	Discretelegend
5	Layout	Single cell: Layout overlay Multi cell: Layout datalattice; Layout datapanel

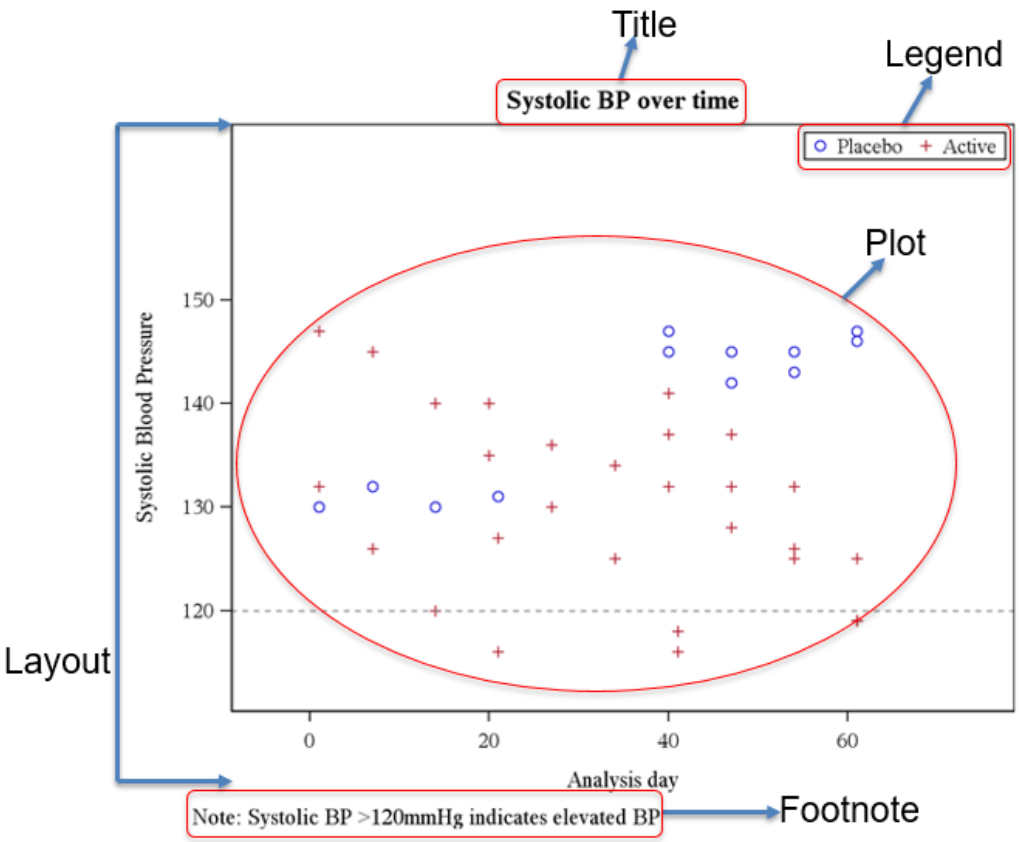


Figure 1. Graph Elements

In creating a SAS graph, we have to know what are all the SAS elements that we want in our graph and how do we want to present them. For example, if we are creating a line plot, 'what' represents to show a line and 'how' represents attributes of this line. E.g. type, color, thickness etc.

Let's expand basic proc template code using below proc template statements. We write all statements in between 'begingraph-endgraph' statements.

```
proc template;
 define statgraph template-name;
 begingraph;
  entrytitle;
  entryfootnote;
  layout overlay/xaxisopts yaxisopts;
    scatterplot x=xvar y=yvar name='scat'; /*example for a plot statement*/
    discretelegend 'scat' /location= inside;
```

```
    endlayout;
  endgraph;
  end;
run;
```

Now, we have the required code to create a template. Let's jump right into creating graphs using dummy data.

1. SCATTER PLOT

This is a plot with duration on x-axis and systolic blood pressure (SBP) on y-axis.

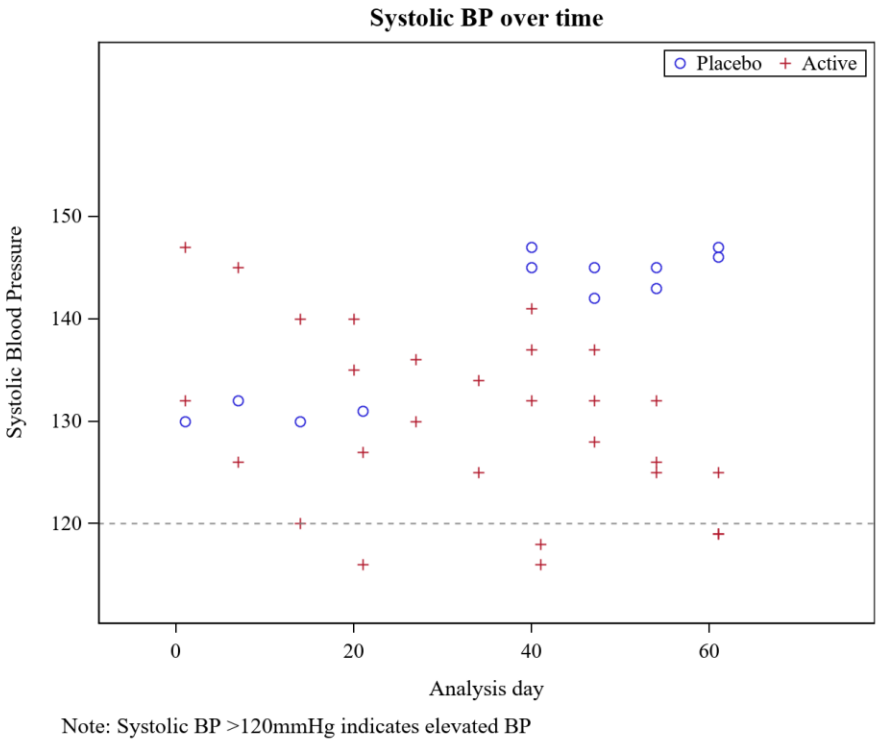


Figure 2. Scatter plot

Step 1. Create template

```
proc template;
   define statgraph scat;
   begingraph;
      entrytitle "Systolic BP over time" ;
      entryfootnote halign=left "          Note: Systolic BP >120mmHg indicates elevated
      BP";
      layout overlay/
      xaxisopts=(offsetmin=0.1 offsetmax=0.2 label= 'Analysis day')
      yaxisopts=(offsetmin=0.1 offsetmax=0.3 label= 'Systolic Blood Pressure');
         scatterplot x=ady y=systolic/group= trtn  name= 'treatment';
         referenceline y=120 /  lineattrs=(pattern=shortdash ) ;
         discretelegend 'treatment' / location= inside  halign=right  valign=top
         border=yes;
      endlayout;
   endgraph;
   end;
run;
```

Step 2: Render the template to the data

```
options orientation=landscape;
ods listing close;
ods rtf file ="pathname/scatter..rtf" ;
ods graphics on/reset=all height=5.5in width=6.5in imagename="scatter"
imagefmt=png noborder ;

proc sgrender data=heart2 template=scat;
format trtn trt.;
run;
ods _all_ close;
ods listing;
```

First, we created the template named 'scat' in proc template procedure and then introduced this 'scat' template in proc sgrender. In order to output the generated graph to rtf, we used SAS output delivery system (ODS). Let's look at some other interesting things in this code.

TITLE AND FOOTNOTES

- If we have to represent a title in two lines, then we write two 'entrytitle' statements.

  E.g. entrytitle "Systolic BP";

  entrytitle "over time";

- Common attributes of titles and footnotes that we look for in creating these graphs are alignment, text size and text wieght.

  E.g. entrytitle textattrs=(size=12pt weight=normal) halign = center 'Systolic BP over time';

Option	Value
HALIGN	CENTER | LEFT | RIGHT
TEXTATTRS	SIZE|WEIGHT|COLOR

LAYOUT

It is a single celled graph. So, we used 'layout overlay' statement. We provided axis options in this 'layout overlay' statement.

X-axis and Y-axis attributes can be manipulated using 'xaxisopts' and 'yaxisopts' for single celled graphs.

OFFSETMIN: gaps at the beginning of the x-axis or y-axis

OFFSETMAX: gaps at the end of the x-axis or y-axis

'Offsetmin' and 'offsetmax' helps us to keep required distance from the plot area to the cell wall.

PLOT

We created the plot using 'scatterplot' statement and assigned a name called 'treatment' to it. This is done to use this name in creating legend for the plot.

'Referenceline' statement creates a line perpendicular to the axis. Attributes of this line are modified using 'lineattrs' option.

LEGEND

We used the name of the scatter plot 'treatment' in 'discretelegend' statement and the position details on where we want the legend to be.

Option	Value
LOCATION	OUTSIDE | INSIDE
HALIGN	CENTER | LEFT | RIGHT
VALIGN	CENTER | TOP | BOTTOM
ACROSS	*Positive- integer* specifies the number of legend entries that are placed horizontally before the next row begins
BORDER	TRUE | FALSE

ODS

By default, ODS graphics is off. We need to set it on to output the graph to desired destination. A few important ODS options are:

Option	Description
RESET	change the values for these options back to their defaults
WIDTH	width of the graph
HEIGHT	height of the graph
IMAGENAME	Filename of the graph
IMAGEFMT	Type of the image E.g. png, jpeg
NOBORDER	Removes the border around the graph

2. HISTOGRAM

This is a frequency distribution plot with systolic blood pressure (SBP) on x-axis and frequency on y-axis.

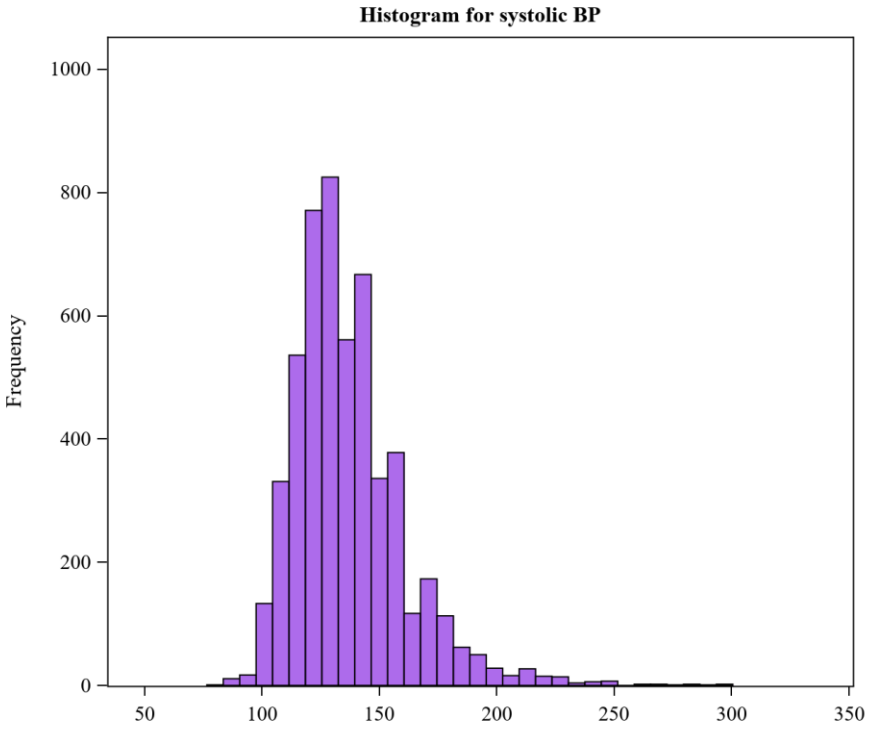


Figure 3. Histogram

```
ods path show;
ods path work.testtemp(update) sashelp.tmplmst(read);
****;
proc template;
   define statgraph histogram/store=work.testtemp;
   begingraph;
      entrytitle textattrs=(size=11pt weight=bold) halign = center 'Histogram for systolic
      BP';
      layout overlay/ xaxisopts=(offsetmin=0.05 linearopts=(viewmin=50   viewmax=350)
      display= (ticks tickvalues))
      yaxisopts=(offsetmax=0.05 linearopts=(viewmin=0 viewmax=1000) label= "Frequency"
      display= (ticks tickvalues label));
         histogram  systolic/ scale=count binwidth= 7 fillattrs=(color=CX8A2BE2
         transparency=0.3 ) ;
      endlayout;
   endgraph;
   end;
run;
```

LAYOUT

1. There are four kinds of axes based on data type. As both x-axis and y-axis have numeric data, we used 'linearopts' in creating this histogram.

Axis type	Option	Data type
Linear	LINEAROPTS	Numeric
Discrete	DISCRETEOPTS	Discrete E.g. Male, Female
Log	LOGOPTS	Logarithmic
Time	TIMEOPTS	SAS time, SAS date, or SAS datetime

2. DISPLAY: controls X, X2, Y, Y2 axis features (LABEL, LINE, TICKS, and TICKVALUES) to be displayed on the primary axis.

Option	Description
DISPLAY= NONE	specifies that no axis features are displayed
DISPLAY= ALL	displays all 4 features
Display = (*list the features*)	select what we want E.g. Display = (ticks tickvalues)

PLOT

In 'histogram' statement we used two options – scale, binwidth. Color and transparency of the bins are adjusted using 'fillattrs' option.

Option	Description
Scale	specifies whether the y-axis displays the frequency counts, percentages or proportions. Default one is percentages
Nbins	Specifies the number of bins.
binwidth	Specifies the width of each bin. If both 'nbins' and 'bindwidth' are used, 'bindwidth' options will be ignored.

ODS

By default, templates created using proc template are stored in sashelp.tmplmst. Sometimes, when we run the GTL code, we might get an error saying:

ERROR: Template 'xxx' was unable to write to template store!

To avoid this, below highlighted statements may be required to store and update templates.

```
ods path show;
ods path work.testtemp(update) sashelp.tmplmst(read);

proc template;
    define statgraph histogram/store=work.testtemp;
```

3. LINE PLOT WITH ERROR BARS

This is a line plot for coagulation parameter values at each visit.

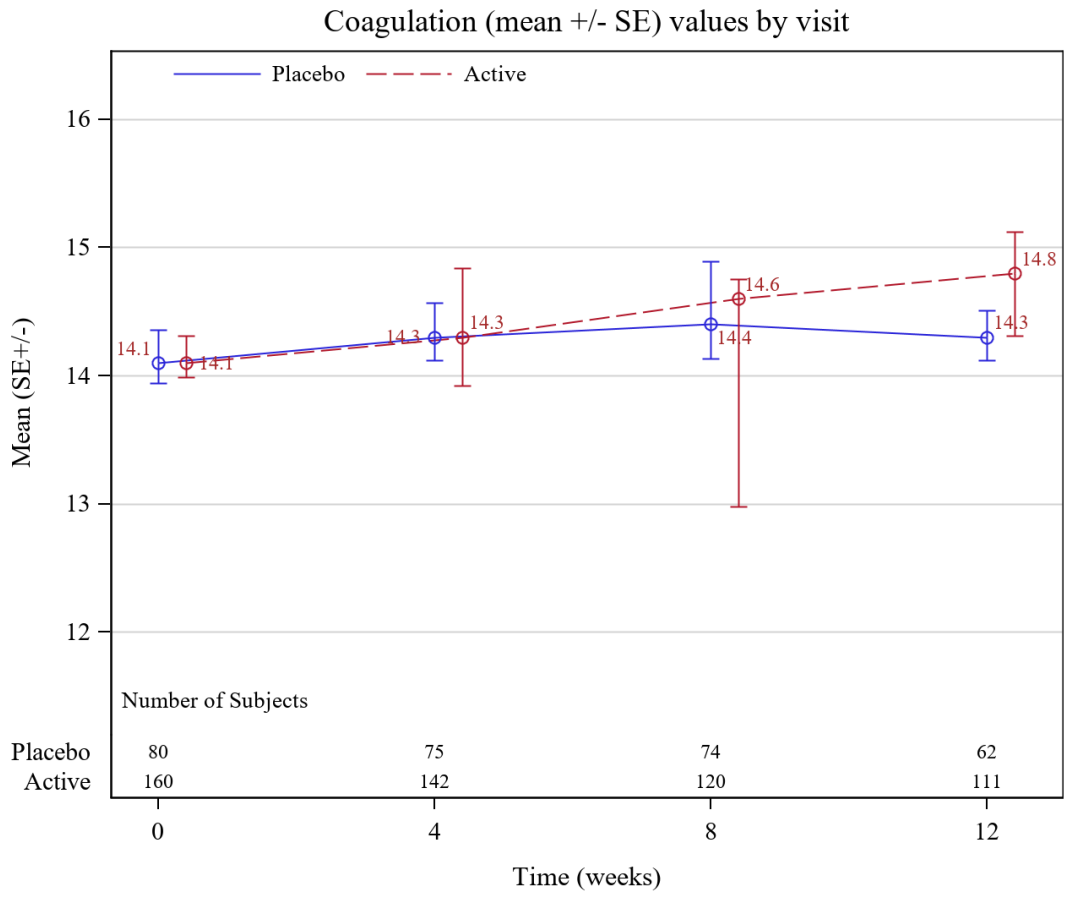


Figure 4. Line plot with error bars

```
proc template;
   define statgraph mean_se;
   begingraph;
      entrytitle textattrs=(size=12pt weight=normal) halign = center 'Coagulation (mean +/-
      SE) values by visit';
      layout overlay/xaxisopts=(offsetmin=0.05 offsetmax=0.05 linearopts=(tickvaluelist= (0
      4 8 12)) label= 'Time (weeks)')
      yaxisopts=(griddisplay=on gridattrs=(thickness= 0.1 color=lightgrey)
      linearopts=(tickvaluepriority=true TICKVALUESEQUENCE=(START=12 END=16 INCREMENT=1))
```

```
      offsetmin=0.15 offsetmax=0.1 label='Mean (SE+/-)');
         seriesplot x= avisitn2 y= mean /group =treatn name= 'trt' ;
         scatterplot x=avisitn2 y=mean/ group=treatn yerrorlower= minus_se
         yerrorupper=plus_se markerattrs=(symbol= graphdata4)
         datalabel= mean datalabelattrs=(color=brown);
         drawtext textattrs=(size=9pt) "Number of Subjects" /anchor=bottomleft width=22
         widthunit=percent
         xspace=wallpercent yspace=wallpercent x=1 y=11 justify=center;
         innermargin/align=bottom pad=0.8;
         axistable x=avisitn value=nsubj / class=treatn   ;
         endinnermargin;
         discretelegend 'trt' / title= " " titleattrs= (size=9pt weight=normal ) location=
         inside  halign=left valign=top
         valueattrs= (size=9pt)  border=false;
      endlayout;
   endgraph;
   end;
run;
```

LAYOUT

1. Grid is shown on y-axis using 'griddisplay' option.
2. There are many ways to provide tick values. We should choose the one that fits our data. Some important tick value options under 'linearopts' are:

Option	Description
TICKVALUELIST	Specifies the order of the tick values for a linear axis as list.
TICKVALUEROTATION	Specifies how the tick values are rotated on the X and X2 axes.
TICKVALUESEQUENCE	Specifies the tick values for a linear axis by start, end, and increment.
VIEWMAX	Specifies the maximum data value to include in the display.
VIEWMIN	Specifies the minimum data value to include in the display.
TICKVALUEPRIORITY	Specifies whether an axis tick specification can extend the axis data range.

We may have to use these options individually or in a combination.

To extend the axis range from the available data values, we can use 'tickvaluepriority=true' in combination with 'TICKVALUESEQUENCE' or 'TICKVALUELIST'

Eg 1: tickvaluepriority=true TICKVALUESEQUENCE= (START=12 END=16 INCREMENT=1)

Eg 2: tickvaluepriority=true TICKVALUELIST= (12 13 14 15 16)

If there are many tick values in an axis which either looks too close or some of tick values are missing on the axis, then you either use TICKVALUEROTATION to present values at an angle which provides space or we can decrease of tick values (E.g. tickvalueattrs=(size=7)) or a combination of both.

PLOT

We can create more than 1 plot in a LAYOUT OVERLAY statement. In this graph, we created line plot using 'seriesplot' and scatter plot with error bars using 'scatterplot'.

Mean values are shown using 'datalabel= mean' in scatter plot statement.

Marker symbols in scatter plot can be modified to our requirement from the default value using 'markerattrs=(symbol=)';

AXIS TABLE

Number of subjects at each time point are shown on x-axis using 'axistable'. It can also be used on X2/Y/Y2 axis. 'Innermargin - endinnermargin' block is used to present axis table within a cell. If we want to present this outside the wall, then

we may have to produce multi-cell graph where the 'axistable' is drawn in a new cell. In this paper, presence of axis table outside the cell wall is explained in Panel box plot.

4. FOREST PLOT:

Displays estimated results to find the effectiveness of the drug.

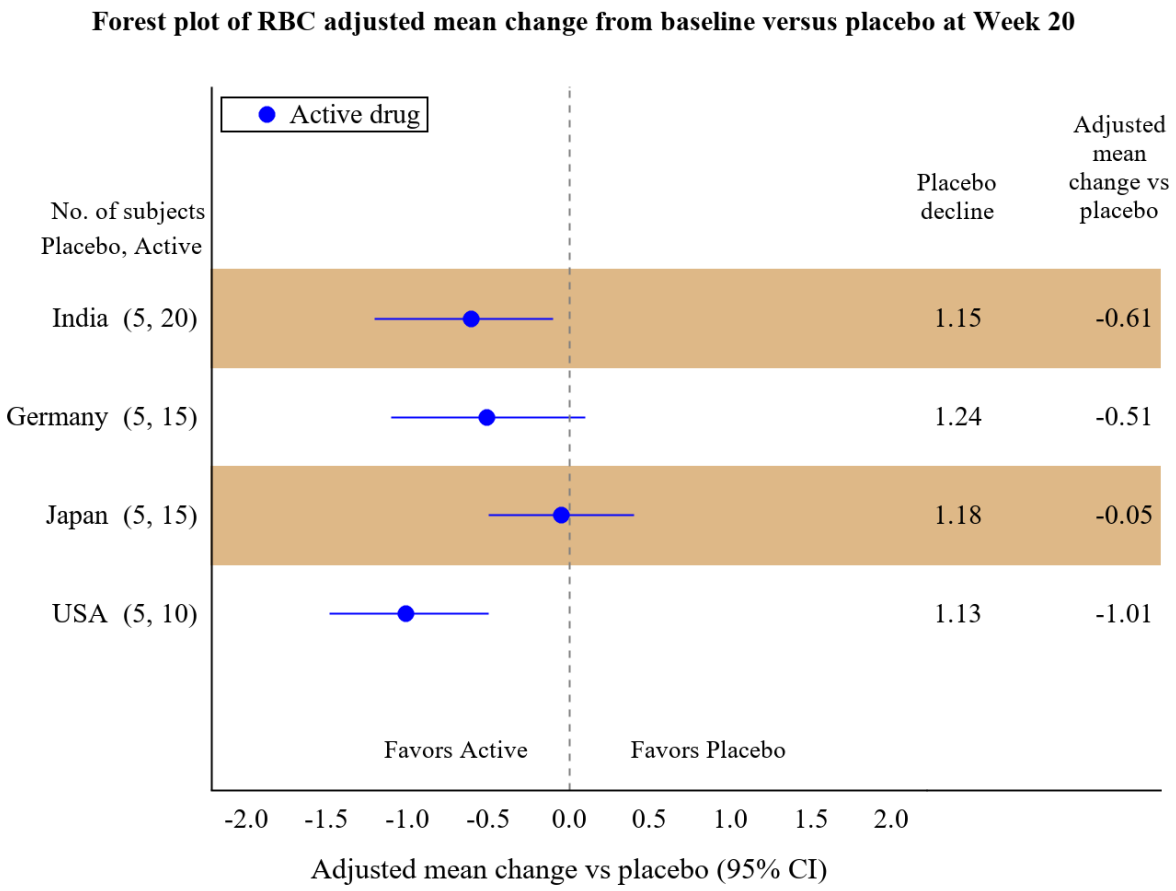


Figure 5. Forest plot

```
proc template;
   define statgraph forest;
   begingraph;
      entrytitle textattrs=(size=10.9pt weight=bold) halign = center   "Forest plot of RBC
      adjusted mean change from baseline versus placebo at Week 20"  ;
      entrytitle  " "  ;
      layout overlay /walldisplay= none xaxisopts=(label="Adjusted mean change vs placebo
      (95% CI)" offsetmin= 0.05 offsetmax=0.05 display= (label tickvalues line)
      linearopts=(tickvaluepriority=true tickvaluesequence=(start=-2 end=2 increment=0.5)))
      yaxisopts=(offsetmin=0.25 offsetmax=0.33 display= (tickvalues line)
      discreteopts=(colorbands=even COLORBANDSATTRS=(color=burlywood)));
         scatterplot x=df_amean y=label /  legendlabel="Active drug"  name= "trt"
         ERRORBARCAPSHAPE= NONE
         xerrorlower=LCL xerrorupper=UCL errorbarattrs=(color=blue)
            markerattrs=(symbol=circlefilled size=8 color=blue );
         referenceline x=0 / lineattrs= ( pattern=2)  ;
         innermargin /align=right;
         axistable y=start value=pdecl / valueattrs=(size=10) display=(values);
         axistable y=start value=empty / valueattrs=(size=10 ) display=(values)
         DATATRANSPARENCY =1;
```

```
        axistable y=start value=df_amean / valueattrs=(size=10) display=(values) ;

        endinnermargin;
        drawtext textattrs=( size=9pt) "Favors Active" /anchor=bottomleft width=18
        widthunit=percent
        xspace=wallpercent yspace=wallpercent x=18 y=3.5 justify=center ;
        drawtext textattrs=( size=9pt) "Favors Placebo" /anchor=bottomleft width=18
        widthunit=percent
        xspace=wallpercent yspace=wallpercent x=44 y=3.5 justify=center ;
        drawtext textattrs=( size=9pt) "No. of subjects" /anchor=bottomleft width=18
        widthunit=percent
        xspace=wallpercent yspace=wallpercent x=-17 y=80 justify=center ;
        drawtext textattrs=( size=9pt) "Placebo, Active" /anchor=bottomleft width=35
        widthunit=percent
        xspace=wallpercent yspace=wallpercent x=-18 y=75 justify=center ;
        drawtext textattrs=( size=9pt) "Placebo decline" /anchor=bottomleft width=10
        widthunit=percent
        xspace=wallpercent yspace=wallpercent x=74 y=80 justify=center ;
        drawtext textattrs=( size=9pt) "Adjusted mean change vs placebo"
        /anchor=bottomleft width=12 widthunit=percent
        xspace=wallpercent yspace=wallpercent x=90 y=80 justify=center;
        discretelegend  'trt' /pad=(left=15 ) LOCATION=INSIDE HALIGN=LEFT VALIGN= TOP
        border=true;
      endlayout;
    endgraph;
    end;
run;
```

LAYOUT

1. To present just the x-axis and y-axis lines and not have a cell wall around the plot, we did the following.
   a. we removed the wall using 'walldisplay= none'
   b. displayed x-axis and y-axis lines using 'display= (line)'
2. Bands on y-axis are shown on even numbers using 'discreteopts=(colorbands=even)'.

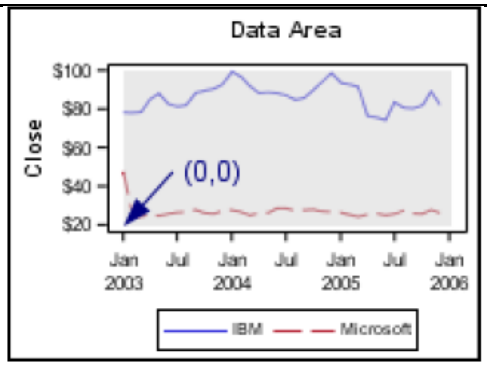
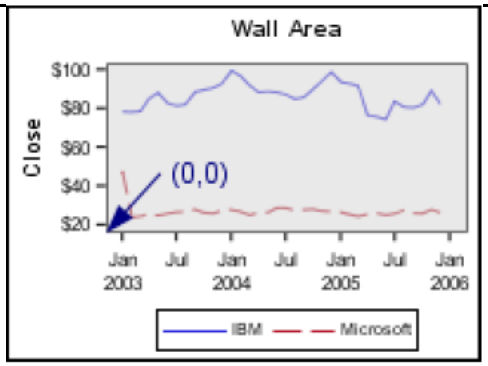
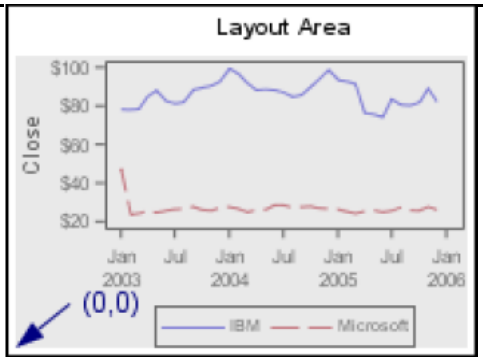
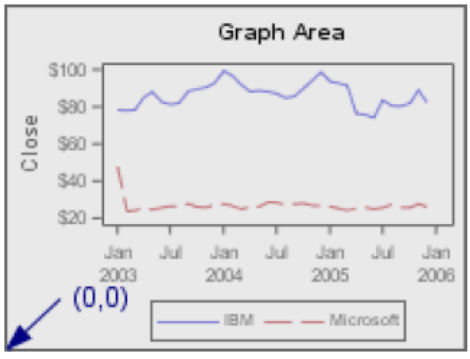
PLOT

By default, error bars in scatterplot has caps. In order to remove them we need to use 'ERRORBARCAPSHAPE= NONE'.

AXIS TABLE

Inferential statistics called 'placebo decline' and 'Adjusted mean change vs Placebo' are displayed vertically along y-axis and inside the layout wall. As these axis tables are inside the wall, we used 'innermargin-endinnermargin' block in the code. In the previous graph (line plot with error bars), axis table is mentioned on x-axis and inside the layout wall.

TYPES OF DRAWING SPACES IN A GRAPH

Drawing space	Description	Shaded area
Data area	Honors the offsets set for axes	
Wall area	Ignores the offsets set for axes	
Layout area	Graph area excluding title and footnote space	
Graph area	Entire area available for graph display	

ANNOTATIONS

To add custom text or shapes in the graph, we can use GTL annotation facility. There are 2 ways to do annotation in GTL.

1. Draw statements

These are written in proc template code within 'layoutoverlay – endlayout' block. There is no need to write 'annotate' statement in proc template or sganno option in proc sgrender in order to use these statements.

In this paper, Line plot with error bars and Forest plot shows how to use create custom text suing 'drawtext' statement.

E.g. Drawtext, Drawline, Drawrectangle

2. Data-Set-Driven Annotations

2.1 SGANNO macros

These are a bunch of inbuilt macros which helps in creating annotate dataset.

%sganno macro: In order to utilize these macros, first we need to run %sagnno which compiles the other %sganno macros and makes them available.

%sganno_help: This macro helps to understand the purpose of these macros, macro variables and the possible values of macro-variables.

E.g. %sganno_help(sgtext)

A few of %sganno macros to create text or shapes are as follows:

Option	Description
%SGTEXT	Creates an observation that draws a single line of text or the first line of text in a multiline annotation.
%SGLINE	Creates an observation that draws a line.
%SGOVAL	Creates an observation that draws an oval.

Steps for annotation

1. Utilizing these macros, we can create a dataset with instructions to write various texts and shapes.

   ```
   %sganno /*compiles the sganno macros*/
   data anno;
      %sgtext(label="Placebo",x1=5,y1=4,drawspace=" DATAVALUE");
   run;
   ```

2. Write 'annotate' statement in the proc template procedure.
3. Introduce the annotate dataset in proc sgrender sganno option.

   E.g. proc sgrender data=*dataset-name* sganno=*annotate-dataset*; run;

2.2 Annotation variables

1. Create annotate dataset using required variables.

   Some of the important variables are:

   Function: declare what you what to create. E.g. Text, line

   X1: X coordinate values for a given function.

   X1SPACE: drawing space for x coordinate.

   Other attributes like color, size, thickness varies from what kind of text or shapes that we require.
2. Write 'annotate' statement in the proc template procedure.
3. Introduce the annotate dataset in proc sgrender sganno option.

E.g. In this paper, we created dataset driven annotate dataset using annotation variables for KM plot

Three important things in creating annotate dataset using either %sganno macros or annotation variables are:

a. What you want to create. E.g. text, shape

b. Choose drawing spaces for x and y coordinates. E.g. x1space= datavalue y1space=graphpercent

c. Providing coordinates. E.g. x1, y1

For a beginner, it is better to create a annotate dataset using %sganno macros rather than using annotation variables as %sganno_help provides the list, purpose and possible values of macro variables of each %sganno macro.

5. PANEL BAR CHART

It is bar chart where incidence of adverse events is plotted against different treatments and severity at each visit.

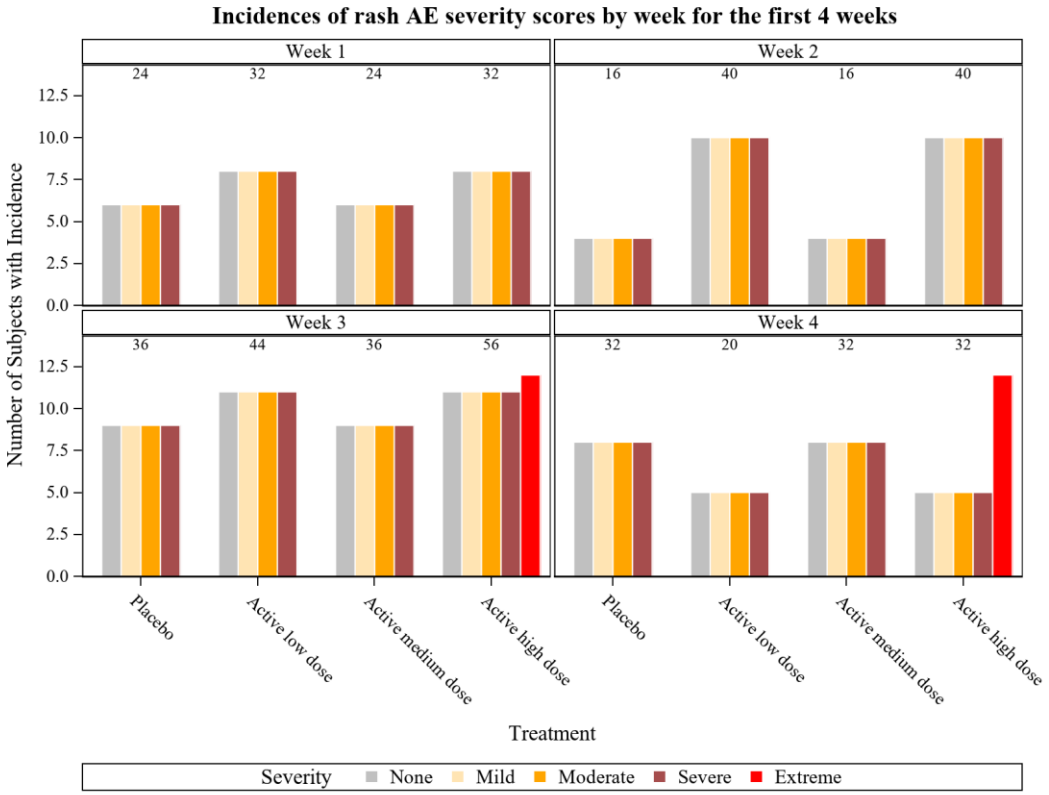


Figure 6. Panel bar chart

```
proc template;
   define statgraph panelchart;
   begingraph;
      discreteattrmap name='Sev';
      value 'None' / fillattrs= (color=grey transparency=0.5) lineattrs=(color=white);
      value 'Mild' / fillattrs= (color= Orange transparency=0.7) lineattrs=(color=white);
      value 'Moderate' / fillattrs= (color= Orange transparency=0) lineattrs=(color=white);
      value 'Severe' / fillattrs= (color= maroon transparency=0.3) lineattrs=(color=white);
      value 'Extreme' / fillattrs= (color= red transparency=0) lineattrs=(color=white);
      enddiscreteattrmap;
      discreteattrvar attrvar=max_sev_map var=aesev attrmap='Sev';
```

```
      entrytitle 'Incidences of rash AE severity scores by week for the first 4 weeks';
      layout datapanel classvars=(avisitn) /  headerlabeldisplay= value  columns=2
      columngutter=2  rowgutter=2 columndatarange=union columnweight=proportional
      columnaxisopts=(label = "Treatment"  tickvalueattrs=(size=9) )
      rowaxisopts=(display= (label ticks tickvalues) offsetmin=0
      label = "Number of Subjects with Incidence "  tickvalueattrs=(size=9));
         layout prototype;
            barchart x=trt  y=nsubj2/ group=max_sev_map name='bar' groupdisplay =cluster
            grouporder=ascending;
            innermargin/align=top;
            axistable x=trt value=nsubj / display=(values);
            endinnermargin;
         endlayout;
         sidebar ;
         discretelegend 'bar' /  title= "Severity";
         Endsidebar;
      endlayout;
   endgraph;
   end;
run;
```

ATTRIBUTES

The plot attributes can be modified at various levels.

1. Discreteattrmap

It is used to create an attribute map for all possible values of a grouping variable. There are 4 types of attributes. They are lineattrs, markerattrs, textattrs and fillattrs. In this barchat (panel) graph, fillattrs and lineattrs of bars are updated from their default values using discreattrmap. Once the map is created, it is applied to the data grouping variable using 'discreteattrvar' statement to create a new mapped variable under 'attrvar=' option. This new mapped variable is used in place of data grouping variable wherever necessary in order to get the required attributes.

Discreteattrmap concept is used in this bar chart to provide the fill colors and line colors for all possible values of adverse event severity.

2. Custom style

Each output delivery system (ODS) destination has a default style. We can also create custom style from the available styles or a new style from scratch.

This code provides the list of available styles.

```
proc template;
  list styles;
run;
```

This code provides all the style elements in 'default' style.

```
proc template;
  source styles.default;
run;
```

This code creates a custom style called 'mystyle' where marker symbols are updated from its original value in 'default' parent style. Except for these marker symbols update, all other remaining style elements in 'mystyle' would be same as in 'default' style.

Once we create a custom style, we need to introduce the custom style in ods rtf 'style' option.

```
proc template;
   define style mystyle;
   parent=Styles.Default;
   style GraphData1 from GraphData1/
   markersymbol = "triangle";
   style GraphData2 from GraphData2/
   markersymbol = "square";
```

```
   end;
run;
ods rtf file = "pathname/plotname.rtf" style=mystyle;
```

Besides updating the plot attributes, we can change many other aspects of graph like attributes of title, footnote, background color etc. This is a complex method compared to other methods and may be used only when necessary.

3. Plot level

We can update the default attributes at plot level. In this paper, forest plot and line plot with error bars attributes are updated in this way. In line plot with error bars, symbol is updated to circle from its default value. It ignores grouping variable and provides the same symbol for all the values of grouping variable. If we want a different set of attributes for each value of grouping variable, then we can use 'discretattrmap'.

E.g. scatterplot x=avisitn2 y=mean/ group=treatn markerattrs=(symbol= circle);

4. Begingraph options

Attributes can also be changed in Begingraph statement by using options.

Option	Description
DATACOLORS	Specifies the list of colors for fill space
DATACONTRASTCOLORS	Specifies the list of contrast colors for lines and markers
DATAFILLPATTERNS	Specifies the list of fill patterns
DATALINEPATTERNS	Specifies the list of line patterns
DATASYMBOLS	Specifies the list of marker symbols

```
proc template;

 define statgraph panelchart;
 begingraph/ datacolors=(red orange yellow)
                 datacontrastcolors=(black blue red);
                 datafillpatterns=(L1 L3 L5)
                 datalinepatterns=(solid shortdash LongDash)
                 datasymbols=(circle triangle square);
 ...GTL statements...;
 endgraph;
 end;
run;
```

The list of colors, symbols and patterns provided using this approach apply to the data in the order of their appearance in the dataset.

For example, if we use 'datacontrastcolors=(black blue red)' to identify mild, moderate and severe events in a study with two treatment groups A and B. If treatment A has mild, moderate and severe adverse events and treatment B has only moderate and severe events, then treatment B moderate events and treatment A mild will have black colored attributes which may not be desirable. So, if the data do not have all possible values for grouping variables under each treatment group, 'discreteattrmap' would be a better option to assign proper colors.

LAYOUT (MULTI CELL)

Single celled graphs are usually created using 'layout overlay'. One way to create a multicell graph is by using 'layout datapanel'. In this statement we specify the number of rows and columns using options 'rows=' and 'columns='. 'Classvar=' option is used to specify the list of classification variable. Each 'classvar=' variable value refers to a cell in the mutli cell graph.

Layout prototype: It is similar to 'layout overlay' statement which is used in 'layout datapanel'.

Axis table is created inside the wall at the top using 'axistable' statement in 'innermargin-endinnermargin' block.

AXES

'Columnaxis' refers to x-axis for all cells and 'rowaxis' refers to y-axis for all cells

LEGEND

Sidebar: It spans the legend across the cells.

6. PANEL BOX PLOT

It is a panel box plot for SBP and Diastolic Blood Pressure (DBP) grouped by 3 categorical variables – study, race and sex.

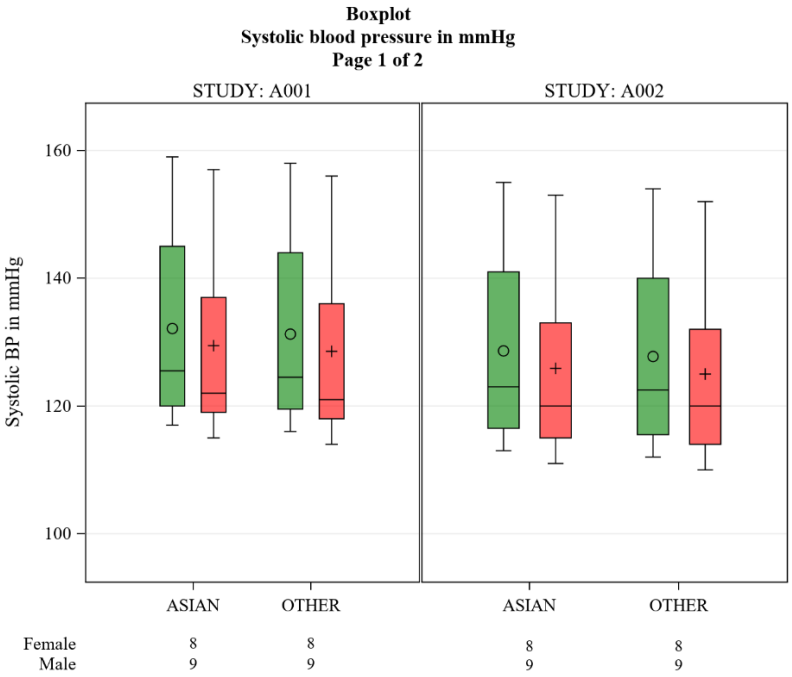


Figure 7. Panel box plot (page 1)

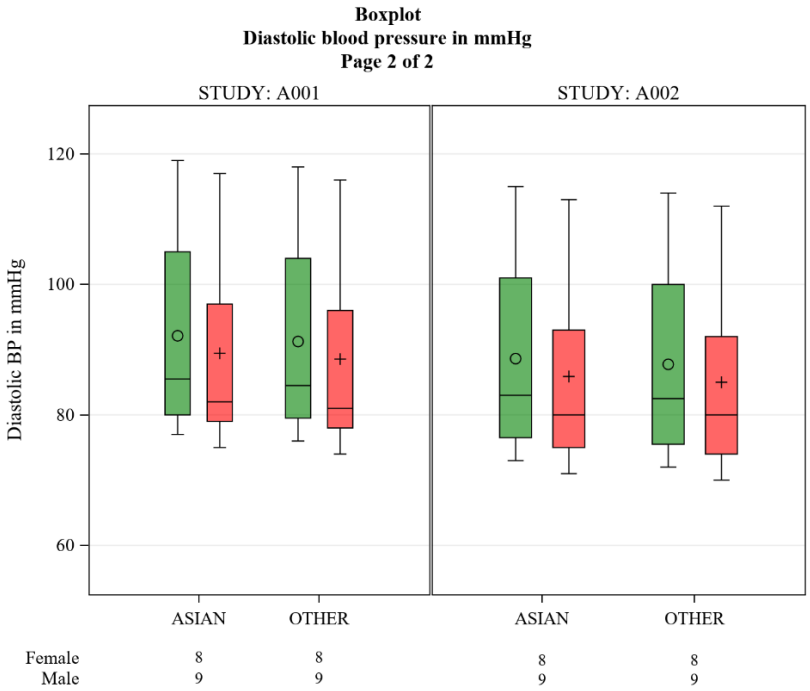


Figure 8. Panel box plot (page 2)

```
proc template;
   define statgraph box_panel;
   dynamic _BYVAL_ _ylabel _ticks _pg;
   begingraph;
      discreteattrmap name='Sex';
      value 'Female' /markerattrs=(color=black) textattrs= (color=green) fillattrs=
      (color=green transparency=0.4) lineattrs=(color=black);
      value 'Male' / markerattrs=(color=black) textattrs= (color=red ) fillattrs= (color=
      red transparency=0.4) lineattrs=(color=black);
      enddiscreteattrmap;
      discreteattrvar attrvar=sex_map1 var=sex_a1  attrmap='Sex';
      discreteattrvar attrvar=sex_map2 var=sex_a2 attrmap='Sex';
      entrytitle textattrs=(size=11pt weight=bold ) halign = center 'Boxplot ';
      entrytitle textattrs=(size=11pt weight=bold) halign = center _BYVAL_;
      entrytitle textattrs=(size=11pt weight=bold) halign = center _pg;
      layout lattice / columns=2 rows=2 columndatarange=unionall rowdatarange=unionall
      rowweights= (0.9 0.1) columnweights=(0.5 0.5);
         rowaxes;
         rowaxis / offsetmin=0.1 offsetmax=0.1 griddisplay=on display=(ticks tickvalues
         label) label=_ylabel type=linear
         linearopts=(tickvaluelist=_ticks tickvaluepriority=true) ;
         rowaxis / display=none ;
         endrowaxes;
         column2headers;
         entry textattrs=graphlabeltext(weight=normal) 'STUDY: A001';
         entry textattrs=graphlabeltext(weight=normal) 'STUDY: A002';
         endcolumn2headers;
         layout overlay / xaxisopts=(display=(ticks tickvalues))
             x2axisopts=(display=(label) );
            boxplot x=race_a1 y=aval_a1/group=sex_map1 groupdisplay=cluster ;
         endlayout;
         layout overlay / xaxisopts=(display=(ticks tickvalues)) ;
            boxplot  x=race_a2 y=aval_a2  /group=sex_map2 groupdisplay=cluster;
         endlayout;
         Layout Overlay / walldisplay=none xaxisopts=(display=none griddisplay=off
         displaySecondary=none)
         x2axisopts=(display=none griddisplay=off displaySecondary=none);
            AxisTable Value=nsubj_a1 X=race_a1 /class=sex_a1 labelPosition=min
            ValueAttrs=(size=9 ) Display=(Label );
         endlayout;
         Layout Overlay / walldisplay=none xaxisopts=(display=none griddisplay=off
         displaySecondary=none)
         x2axisopts=(display=none griddisplay=off displaySecondary=none);
            AxisTable Value=nsubj_a2 X=race_a2/class=sex_a2  labelposition=max labelattrs=
            (color=white size=0) ValueAttrs=(size=9 );
         endlayout;
      endlayout;
   endgraph;
   end;
run;

ods listing close;
%macro boxp;
   ods rtf file="pathname\custom_panel_bar_plot.rtf" ;
   options orientation=landscape ;
   %do i = 1 %to 2;
   ods graphics on/reset=all height=6.5in width=8.5in imagename="boxpanel" imagefmt=png
   noborder;
   proc sgrender data=boxd5 template=box_panel;
      dynamic %if &i=1 %then %do;
      %str(_ylabel= "Systolic BP in mmHg")
      %str(_ticks="100 120 140 160 ")
      %str(_pg="Page 1 of 2");
```

```
        %end;
        %else %if &i=2 %then %do;
        %str(_ylabel= "Diastolic BP in mmHg")
        %str(_ticks="60 80 100 120 ")
        %str(_pg="Page 2 of 2");
        %end;
        format  pg byval.;
        where pg = &i;
        by pg;
      run;
      %end;
   %mend;
   %boxp
   ods _all_ close;
   ods listing;
```

LAYOUT (MULTICELL)

It is a multi-cell graph with 2 rows and 2 columns. We used 'layout lattice' instead of 'layout datapanel' in creating this graph. It is because, in previous panel bar plot, all 4 cells have same information i.e., bar plots. But here, two cells have box plots and other two cells have number of subjects' information. By using 'layout lattice' we can create different plots in each cell.

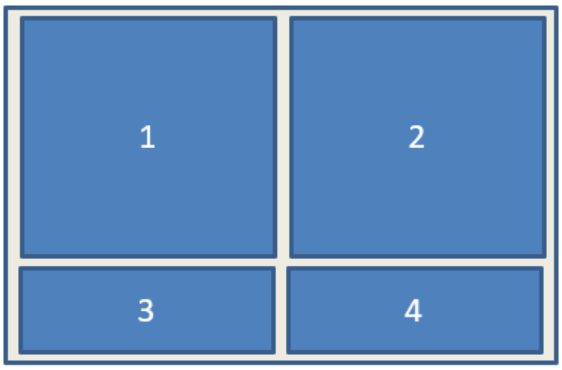


Figure 9. Multi cell layout

```
layout lattice / columns=2 rows=2 rowweights= (0.9 0.1) columnweights=(0.5 0.5);
  layout overlay ; /*1st cell plot*/
    boxplot  ;
  endlayout;
  layout overlay  ; /*2nd cell plot*/
    boxplot  ;
  endlayout;
  Layout Overlay / walldisplay=none; /*3rd cell plot*/
    AxisTable ;
  endlayout;
  Layout Overlay / walldisplay=none; /*4th cell plot*/
    AxisTable  ;
  endlayout;
endlayout;
```

Figure 10. Code for multi cell layout

Under 'layout lattice', there are 4 'layout overlay' statements. Each 'layout overlay' refers to a cell. Cells 3 and 4 in Figure 7 and Figure 8 may not appear as a cells by themselves because cell wall is removed by using 'walldisplay = none' option. To display an axis table inside the cell wall as shown in Figure 4, Figure 5 and Figure 6, we can use 'innermargein – endinnermargin' block and there is no need to create a separate cell for it. But, to create an 'axistable' outside the cell wall, we may need to create a multi-cell graph as shown in this box plot.

DYNAMIC VARIABLES

Proc template allows us to write flexible code with variables called dynamic variables. As a result, we can define the values for these variables in the execution phase instead of compilation phase. First, we need to declare these variable names under 'dynamic' statement in proc template and define the dynamic variable values in proc sgrender as shown in Figure 11. Dynamic variables have many advantages. In this boxplot, we needed different y-axis label and tick values in each page. So, we declared these as dynamic variables and defined the values during execution in proc sgrender.

There are several special predefined dynamic variables that we can use in GTL. These special variables are called special dynamic variables. These variables need to be declared in the 'dynamic' statement in proc template but are not required to the defined in 'proc sgrender'. E.g. _byval_.

In this graph, we used '_byval_' in 'entrytitle' statement to get different subtitle for each page. '_byval_' takes the corresponding 'by' variable value in proc sgrender for each page.

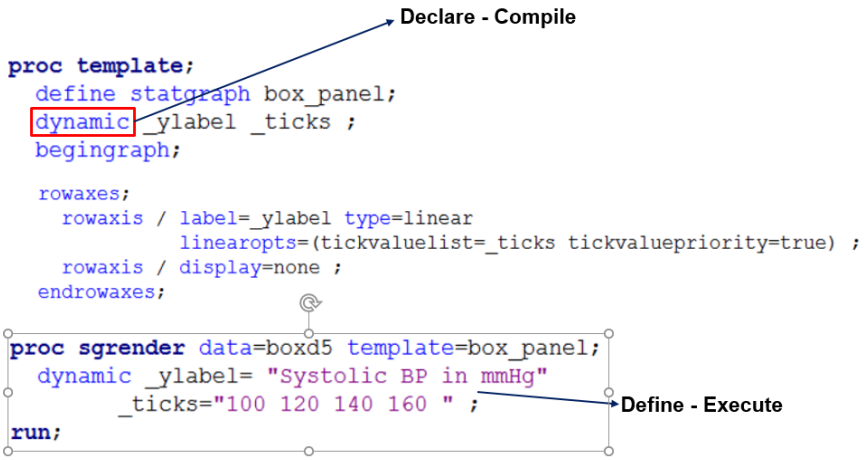


Figure 11. Dynamic variables

CONDITIONAL LOGIC

GTL allows to use conditional statements in proc template. It is of great use to adjust or change content from one page to another in a multiple page graph output. The syntax for conditional logic is shown below.

IF (condition)

GTL statement(s);

ELSE

IF (condition)

GTL statement(s);

ELSE

GTL statement(s);

ENDIF;

ENDIF;

Here is an example for annotating different text in each page of a 2-page graph output.

```
if (_byval_ = "Part 1 of 2: Female")
   drawtext textattrs= (size=8.5 pt) "Absolute risk (95% CI)"/
   anchor=bottomleft width=22 widthunit=percent xspace=wallpercent yspace=wallpercent x=82
   y=82 justify=center;
else
   drawtext textattrs= (size=8.5 pt) "Relative risk (95% CI)"/
   anchor=bottomleft width=22 widthunit=percent xspace=wallpercent yspace=wallpercent x=82
   y=82 justify=center;
endif;
```

7. WATERFALL PLOT

It is a bar chart which represents response of subjects to a given drug.

This plot is drawn for a cancer study where subject number is represented on x-axis and best percentage change from baseline (PCHG) values are shown on y-axis. PCHG values are grouped to 4 categories. They are:

PD – Progressive disease

SD – Stable disease

PR – Partial response

CR – Complete response

Effectiveness of the drug: CR is the best possible result and PD is the worst possible result.

CR > PR > SD > PD

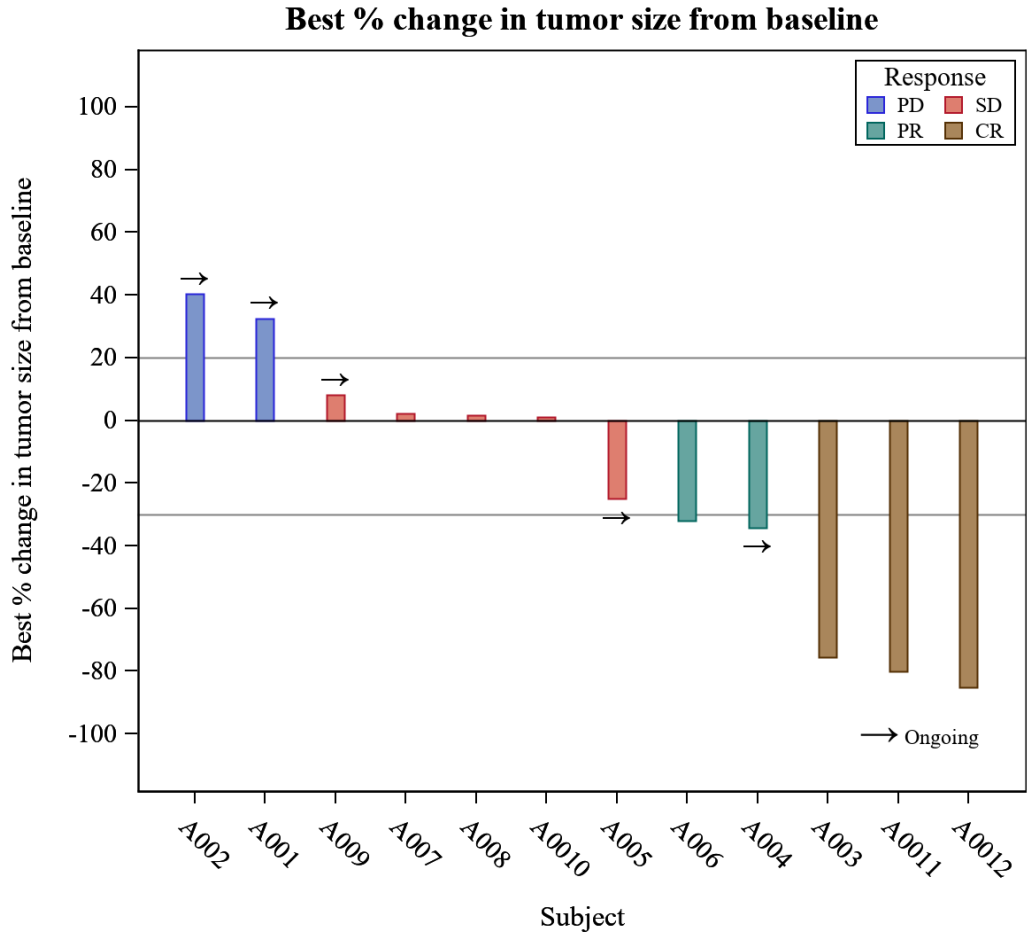


Figure 12. Waterfall plot

```
proc template;
   define statgraph waterfall;
   begingraph;
      entrytitle "Best % change in tumor size from baseline";
      symbolchar name=rightarrow char='2192'x;
      layout overlay/xaxisopts=( label = "Subject" type=discrete)
      yaxisopts =(label = "Best % change in tumor size from baseline"
      LINEAROPTS=(tickvaluepriority=true tickvaluesequence=(start=-100 end=100
      increment=20)));
         referenceline y=20 / lineattrs=(thickness=1.5);
```

```
        referenceline y=-30 / lineattrs=(thickness=1.5);
        barchart x=subjid y=pchg/ group=response barlabelattrs=(size=6pt)  barwidth=0.25
        name="BAR";
        scatterplot x=subjid y=marker / markerattrs=(symbol=rightarrow color=black size=30
        weight=bold);
        discreteLegend "BAR"/ across=2 autoalign=(topright) location=inside
        titleattrs=(size=10pt)
        valueattrs=(size=8pt) border=true borderattrs=(color=black) title="Response";
        drawtext textattrs=(size=15pt color=black weight=bold) {unicode "2192"x} /
        anchor=bottomright width=9 widthunit=percent xspace=wallpercent yspace=wallpercent
        x=86 y=4.75 justify=center ;
        drawtext textattrs=(size=8pt) "Ongoing" / anchor=bottomright width=12
        widthunit=percent xspace=wallpercent yspace=wallpercent x=95 y=5.35 justify=center
        ;
    endlayout;
  endgraph;
  end;
run;
```

SYMBOLCHAR

Defines a marker symbol using a Unicode character that can be referenced in other statements.

We wanted a right arrow as a symbol for subjects who are still ongoing in the study. Right arrow is not directly available in 'scatterplot' statement as an option under 'markerattrs=(symbol=)'. So, we used 'symbolchar' statement to define right arrow and used it in 'scatterplot' statement.

Here is a link to find various Unicode values http://www.unicode.org/charts/charindex.html

8. SPIDER PLOT

It is a series plot which shows how tumor size changed from baseline during the study. Each line represents a subject and the color of the lines represent what kind of response the subject had.

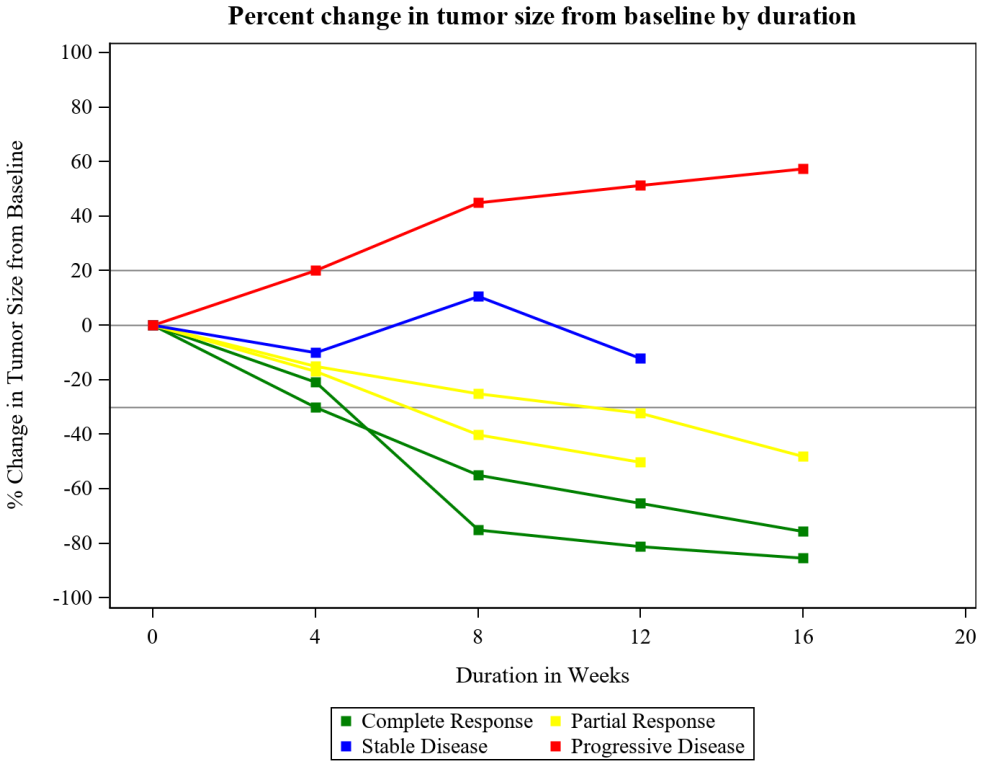


Figure 13. Spider plot

```
proc template;
   define statgraph spider/store=work.testtemp;
   begingraph;
      discreteattrmap name='criteria';
      value 'Progressive Disease' / lineattrs=(color=red) markerattrs=(color=red);
      value 'Complete Response' / lineattrs=(color=green) markerattrs=(color=green);
      value 'Partial Response' / lineattrs=(color=yellow) markerattrs=(color=yellow);
      value 'Stable Disease' / lineattrs=(color=blue) markerattrs=(color=blue);
      enddiscreteattrmap;
      discreteattrvar attrvar=resn_map var=response attrmap='criteria';
      entrytitle "Percent change in tumor size from baseline by duration";
      layout overlay / xaxisopts=(label='Duration in Weeks'
      linearopts=(tickvaluesequence=(start=0 end=20 increment=4)
      viewmin=0 viewmax=20) offsetmin=0.05)
      yaxisopts=(label='% Change in Tumor Size from Baseline'
      linearopts=(tickvaluepriority=true tickvaluesequence=(start=-100 end=100
      increment=20)));
         referenceline y=0/ lineattrs=(thickness=1.5);
         referenceline y=20/ lineattrs=(thickness=1);
         referenceline y=-30/ lineattrs=(thickness=1);
         seriesplot x=avisitn y=pchg/group=id linecolorgroup=resn_map
         lineattrs=(thickness=2 pattern=solid) groupdisplay=overlay break=true;
         scatterplot x=avisitn y=pchg/ group=resn_map markerattrs=(size=6
         symbol=squarefilled) groupdisplay=overlay name='a';
         discretelegend "a"/ valign=bottom across=2;
      endlayout;
   endgraph;
   end;
run;
```

PLOT

Plot is grouped by subject number (group=id) but the color grouping is based on subject response to the drug (linecolorgroup=resn_map). 'Discreteattrmap' is used to define colors for line and marker symbols.

9. SWIMMER PLOT

It is a bar chart to show all the responses of a subject in one glance. Each bar represents a subject's duration in the study. Colored squares on the bars represent various kinds of response that a subject had.

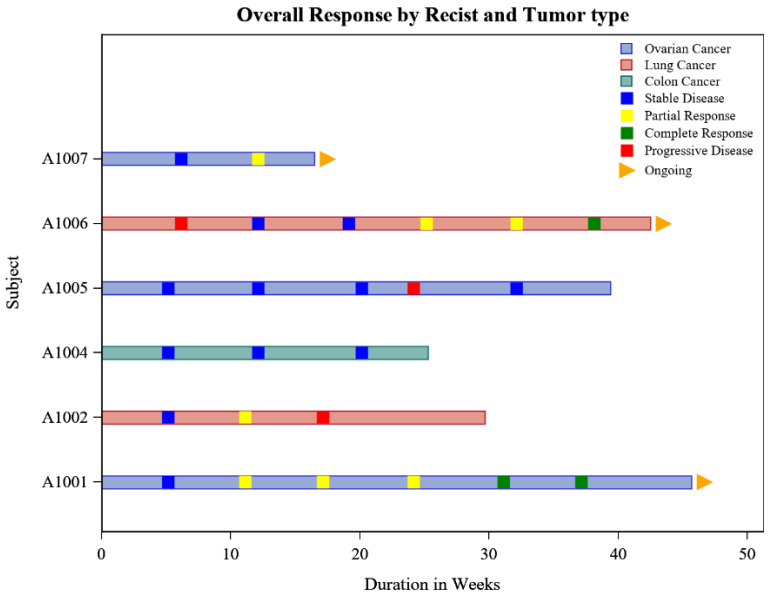


Figure 14. Swimmer plot

```
proc template;
   define statgraph swimmer;
   begingraph;
      discreteattrmap name='restype';
      value 'Stable Disease' / markerattrs=(symbol=squarefilled color=blue size=10);
      value 'Partial Response' / markerattrs=(symbol=squarefilled color=yellow size=10);
      value 'Complete Response' / markerattrs=(symbol=squarefilled color=green size=10);
      value 'Progressive Disease' / markerattrs=(symbol=squarefilled color=red size=10);
      enddiscreteattrmap;
      discreteattrvar attrvar=resp_map var=resp attrmap='restype';
      entrytitle "Overall Response by Recist and Tumor type";
      layout overlay/ xaxisopts=(label='Duration in Weeks' offsetmin=0
      linearopts=(tickvaluesequence=(start=0 end=50 increment=10) viewmin=0 viewmax=50))
      yaxisopts=(label='Subject' offsetmax=0.25);
         barchart category=id1 response=tot_dur1/ group=tumor_type1 orient=horizontal
         barwidth=0.2 name='bar' fillattrs= (transparency=0.2)
         INCLUDEMISSINGGROUP=FALSE;
         scatterplot X=resp_dur  Y=id/group=resp_map name="res";
         scatterplot x=ongdur y=id / markerattrs=(symbol=trianglerightfilled color=orange
         size=13) name='ongo' legendlabel="Ongoing";
         discreteLegend "bar"  "res" "ongo"/across=1 autoalign=(topright)
         location=inside valueattrs=(size=8pt) border=false;
      endlayout;
   endgraph;
   end;
run;
```

PLOT

'Discreteattrmap' is used for drug response where we chose different colored square markers for each kind of response. Bar colors are default colors provided by template based on grouping variable.

There are 3 plots in this graph.

1. Bar chart: It represents duration of each subject on the study
2. Scatter plot: Colored squares represents subjects' response to the drug
3. Scatter plot: Orange triangle represents ongoing subjects in the study

LEGEND

Legend names for all plots are included in one 'discretelegend' statement.

10. KAPLAN MEIER (KM) PLOT

It gives survival or failure probability for subjects based on study treatment.

To create this plot, first we need to create time to event analysis dataset (Table 1). An event could be of many kinds.

E.g. Time to death

Time of first occurrence of Adverse Event

Time to first occurrence of abnormal value of a parameter

Table 1 is an example for time to event analysis dataset. Important variables in this dataset are:

Variable	Description
PARAM	Refers to first abnormal occurrence of RBC value
AVAL	Visit at which the analysis is done
TRTN	Numeric variable for a given treatment
ID	Subject number
CNSR	Differentiates subjects who got the event from who did not. E.g. CNSR=1 refers to subjects who didn't get the event till their latest visit represented in variable AVAL E.g. CNSR=0 refers to subjects who got the event at visit represented in variable AVAL

	PARAM	PARAMCD	AVAL	TRTN	id	cnsr
1	First abnormal RBC value at day	ABN_RBC	60	0	1	1
2	First abnormal RBC value at day	ABN_RBC	40	0	2	0
3	First abnormal RBC value at day	ABN_RBC	40	0	3	1
4	First abnormal RBC value at day	ABN_RBC	40	0	4	0

Table 1. ADTTE dataset

Next, run 'proc lifetest' on ADTTE dataset to get survival probability. Failure probability or probability of progression can be obtained in two ways.

1. Obtain 'survivalplot' statistics from proc lifetest and subtract them from 1.

   E.g. Failure probability = 1 – survival probability

2. Get failure probability directly from 'proc lifetest' by using 'failure' option and output 'failureplot' statistics using ods output.

   E.g. `plots=survival(atrisk=0 20 40 60 failure)`

'Outsurv=dataset' provides estimates with confidence intervals which we presented in the figure at the top using dataset driven annotation.

'Plots=dataset' provides survival probability from which we derived failure probability. This is used to create step plot.

```
 **surviival analysis;
ods listing close;
ods graphics on;
ods output SurvivalPlot= surv_risk;
proc lifetest data= adtte plots=survival(atrisk=0 20 40 60)
           outsurv=surv timelist=(0 20 40 60) reduceout;
    time aval*cnsr(1); /*cnsr(1) represents the censored subjects*/
    strata trtn;
run;
ods _all_ close;
ods listing;
```

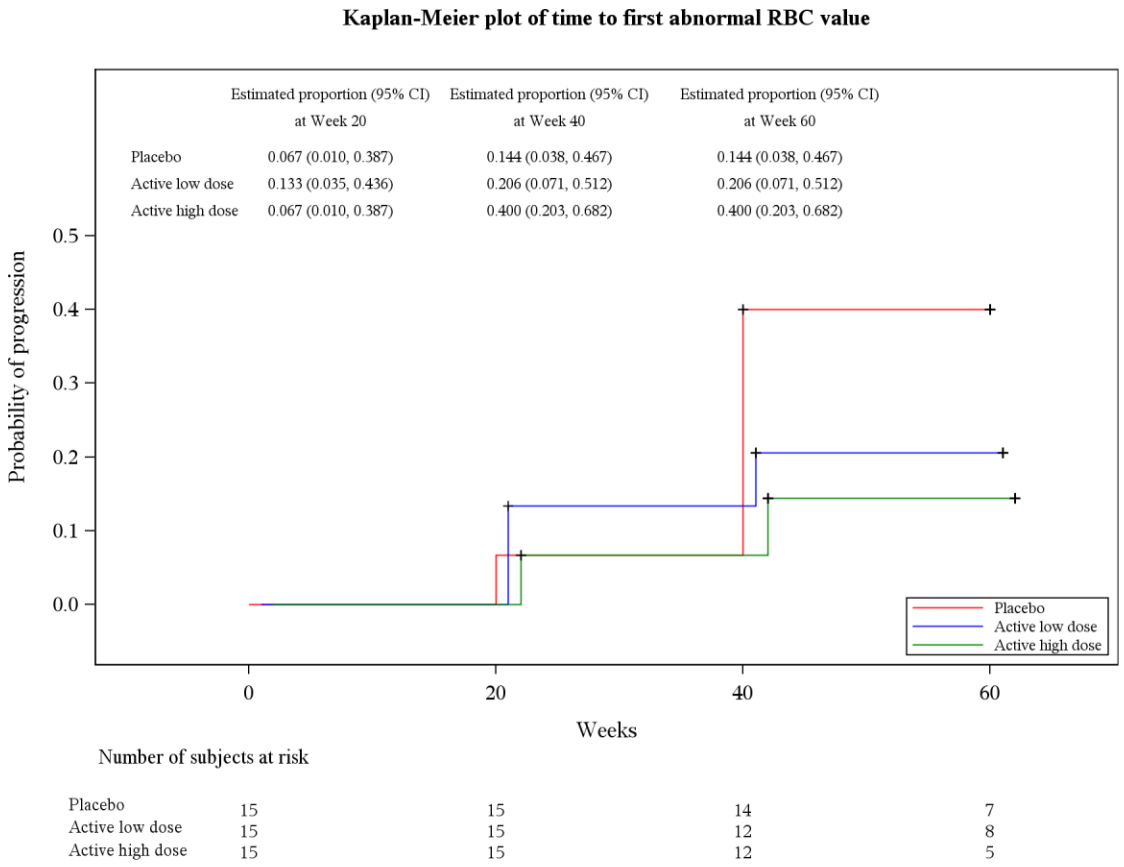


Figure 15. KM plot

```
proc template;
   define statgraph km2/store=work.testtemp;
   begingraph;
      discreteattrmap name='Attr_trt';
         value 'Placebo' / lineattrs=(color=red pattern=solid);
         value 'Active low dose' / lineattrs=(color=blue pattern=solid);
         value 'Active high dose' / lineattrs=(color=green pattern=solid);
      enddiscreteattrmap;
      discreteattrvar attrvar=Attr_trtname var=stratumnum attrmap='Attr_trt';
      entrytitle textattrs=(size=11pt weight=bold) halign = center 'Kaplan-Meier plot of
      time to first abnormal RBC value';
      entrytitle " " ;
      layout  lattice / columns=1 rows=2 rowweights= (0.85 0.15) columndatarange=union;
         /*step plot*/
         layout overlay/xaxisopts=(offsetmin=0.15 offsetmax=0.1
         label = "Weeks"
         linearopts=(tickvaluelist= (0 20 40 60)))
         yaxisopts=(offsetmin=0.1 offsetmax=0.28 label = 'Probability of progression'
         linearopts=(TICKVALUEPRIORITY = true tickvaluesequence=(start=0 end=0.5
         increment=0.1)));
            stepplot x= time y= failure /group=Attr_trtname  name='step' ;
            scatterplot x= time y= f_censored /group=Attr_trtname  name='scatter'
            markerattrs=(symbol=plus color=black) ;
            discretelegend 'step' /location= inside halign=right valign=bottom
            valueattrs=(size=8)  border= yes across=1;
         endlayout;
         /*no.of subjects at bottom presented as plot*/
         Layout Overlay / walldisplay=none xaxisopts=(display=none griddisplay=off
```

```
            displaySecondary=none)
            x2axisopts=(display=none griddisplay=off displaySecondary=none);
               AxisTable Value=atrisk X=tatrisk /class=Attr_trtname  ValueAttrs=( Color=black
               size=9 ) display=(values)
               headerlabel= "Number of subjects at risk" headerlabelattrs=(size=10)
               valuehalign=center;
               drawtext textattrs=( size=9pt) "Placebo" /anchor=bottomleft width=18
               widthunit=percent
               xspace=wallpercent yspace=wallpercent x=-2.7 y=39 justify=center ;
               drawtext textattrs=( size=9pt) "Active low dose" /anchor=bottomleft width=15
               widthunit=percent
               xspace=wallpercent yspace=wallpercent x=-2.7 y=20.5 justify=center ;
               drawtext textattrs=( size=9pt) "Active high dose" /anchor=bottomleft width=18
               widthunit=percent
               xspace=wallpercent yspace=wallpercent x=-2.7 y=1 justify=center ;
            endlayout;
         endlayout;
         annotate;
      endgraph;
      end;
   run;
```

LAYOUT

It is multi-cell plot two cells in it. One cell has step plot, scatterplot and estimates with 95% confidence interval. Another cell shows number of subjects at each visit.

As shown in Figure 16, 'layout lattice' is used to create multi-cell graph. Two 'layout overlay' statements are used within 'layout lattice' for plots within each cell.

```
proc template;
  define statgraph km2;
  begingraph;
    layout lattice / columns=1 rows=2;
      /*step plot*/
      layout overlay;
        ***SAS Statements***;
      endlayout;
      /*no.of subjects at bottom presented as plot*/
      Layout Overlay ;
        ***SAS Statements***;
      endlayout;
     endlayout;
     annotate;
  endgraph;
  end;
run;
```

Figure 16. SAS code for multi cell plot

PLOT

'Stepplot' statement is used to plot failure probability or probability of progression at each visit. 'Scatterplot' statement is to plot censored subject's failure probability at each visit. 'axistable' statement is used to present x-axis table at the bottom of the graph which represents number of subjects at each visit.

ANNOTATION

In this plot, we used dataset driven annotation to show estimates with 95% confidence intervals provided by proc lifetest. Here, we provided x1 and y1 coordinates in 'graphpercent' drawing space in order to place estimates with 95% confidence values right above the step plot.

CONCLUSION

The graphs created in this paper not only helps those who already know how to create graphs using traditional or SG procedures but also to those who are entirely new to graph programming. These plots may also be created in different ways. This is our approach towards creating these graphs based on our knowledge and experience. By using GTL concepts and examples mentioned in this paper, we believe it gives a good idea and strong base on how to create SAS graphs using GTL. The example code and dummy data that are provided in GitHub location helps the programmer to recreate and gain confidence in using GTL.

REFERENCES

MATANGE, S. (2019). GETTING STARTED WITH THE GRAPH TEMPLATE LANGUAGE IN SAS: Examples, tips, and techniques for creating custom graphs. SAS Institute.

SAS Institute Inc. 2016. SAS® 9.4 Graph Template Language: User's Guide, Fifth Edition. Cary, NC: SAS Institute Inc.

Unicode® character name index. (n.d.). Accessed April 4, 2021. http://www.unicode.org/charts/charindex.html

RECOMMENDED READING

- *GETTING STARTED WITH THE GRAPH TEMPLATE LANGUAGE IN SAS: Examples, tips, and techniques for creating custom graphs.*

CONTACT INFORMATION

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

Manohar Modem
Sr. Statistical Programmer
Cytel Inc
1050 Winter St Suite 2700
Waltham, MA / 02451
Email: manohar.modem@gmail.com

Bhavana Bommisetty
Statistical Programmer
Vita Data Sciences
281 Winter St Suite 100
Waltham, MA / 02451
Email: bhavana.bommisetty@gmail.com