PharmaSUG China 2023 - Paper DV-163

GTL or SGPLOT: Which one to choose?

Yan He, Fosun Pharma;

ABSTRACT

During the progress of clinical trial projects, generating statistical graphics is an indispensable part of the process of producing CSR or temporary observational data status. The most commonly used methods in SAS for statistical graphics through ODS include Graph Template Language (GTL) and SG procedures. The GTL uses the PROC TEMPLATE to define the different cells that contain the graphics elements firstly, then uses the PROC SGRENDER to read the template and create the graphics, which codes are complex but flexible. Meanwhile, SG procedures are able to specify and create various graphics directly in the procedure step, which codes are relatively simple but may not be applicable to more complex graphics. In the process of plotting, you might have run into this situation we may encounter the awkward situation that we want to use SGPLOT, but after completing the data set and filling in the PROC SGPLOT parameters, we find some details that SGPLOT cannot adjust. Or, in a time crunch, we choose GTL and spend too long adjusting parameters, only to find out later that SGPLOT can actually meet our graphics needs. In this paper, it compared the application of GTL and the most commonly used SG procedures SGPLOT in various graphics scenarios, and suggested more suitable and convenient options for reference.

INTRODUCTION

Statistical graphic is a significant component of outputting CSR or temporary TFLs to statisticians. Sometimes statisticians can give us enough time to output, for example, when preparing the final CSR of the project, which is used for submissions for drug approvals. Sometimes the need is urgent, such as a statistician is going to a meeting to review the latest objective response rate data from an oncology program, and the output is not used in a formal article, and there is no existing code to run directly. At this point, as a statistical SAS programmer, you come to a crossroads. Which way to choose to produce statistical graphs is more efficient?

Of course, if you are an advanced SAS programmer who is very skilled in graphics, you can choose any drawing methods and draw a picture in very little time. But for junior programmers or those who know little about graphics, the choice of which way to draw can be confusing. At least it is true for me. In addition, it is also important to be efficient in our daily work task. With these concerns, I consult some references to understand the difference between GTL and SGPLOT.

By querying the data, I find it that both GTL and SG procedures use the same underlying rendering system. GTL is a very powerful and flexible plotting system, and GTL can complete all types of graphics. SG procedures contain 3 plotting methods: SGPLOT procedure, SGPANEL procedure, SGSCATTER procedure. SGPLOT procedure is for single-cell graphs, SGPANEL procedure is for classification panel graphs, SGSCATTER procedure is for comparative scatter plots and matrices. Since SGPANEL and SGSCATTER are not involved in the application scope of my usual work at present, this article mainly aims at GTL and SGPLOT comparison.

For SGPLOT, which is the most commonly used SG procedures, the types of graphs it can complete include:

- Basic Plots: scatter and series plots
- Fit and Confidence Plots: regression and loess plots
- Distribution Plots: histograms and box plots
- Categorization Plots: bar charts and dot plots

In clinical trial programs, we usually receive plotting requirements including, but not limited to: line plot, bar chart, swimmer plot, spider plot, waterfall plot, KM plot, forest plot, etc. The following article presents

the differences and advantages of GTL and SGPLOT among line plot, swimmer plot, KM plot and forest plot.

1. LINE PLOT

Scenario simulation: You receive an email asking you to make a line plot of K+ from 2 subjects over 6 access periods. You are asked to reply to this email within 2 hours, as shown in the figure 1. Here, I write SGPLOT and GTL two programs for the same plot.

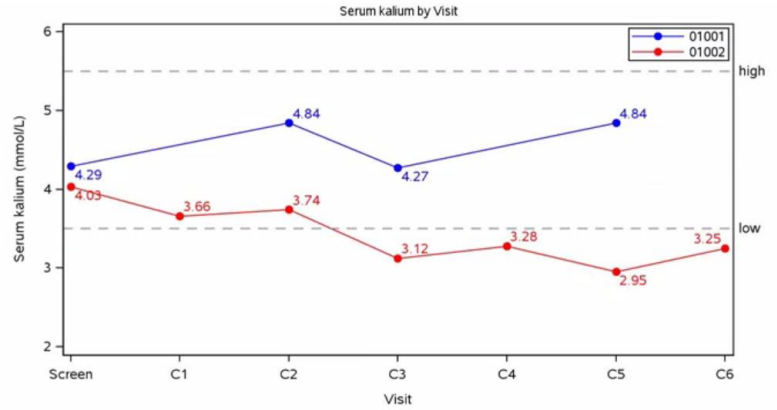

Figure 1. Line plot

SGPLOT code:

```
proc sgplot data=final;
    title "Serum kalium by Visit" ;
    styleattrs datacontrastcolors=(blue red);
    series y=LBSTRESN x=VISIT /GROUP=subject DATALABEL= LBSTRESC
    LINEATTRS=(pattern=Solid) ;
    scatter y=LBSTRESN x=VISIT /GROUP=subject
    MARKERATTRS=(symbol=CircleFilled) ;
    XAxis label="Visit" ;
    YAxis label="Serum kalium (mmol/L)" values=(2 to 6 by 1) ;
    refline 3.5 5.5 / axis=y lineattrs=(pattern=dash color=gray)
    label=("low" "high");
    legenditem type=markerline name='hy1' / label="01001"
    lineattrs=(pattern=Solid color=blue) markerattrs=(symbol=circlefilled
    color=blue);
    legenditem type=markerline name='hy2' / label="01002"
    lineattrs=(pattern=Solid color=red) markerattrs=(symbol=circlefilled
    color=red);
    keylegend "hy1" "hy2"/across=1 location=inside position=topright ;
run;
```

GTL code:

```
proc template;
  define statgraph scatter_series;
    begingraph;
      entrytitle "Serum kalium by Visit";
      discreteattrmap name="subject" / ignorecase=true ;
          value "01001"/markerattrs=(color=blue symbol=CircleFilled)
          lineattrs=(pattern=1 color=blue);
          value "01002"/markerattrs=(color=red symbol=CircleFilled)
          lineattrs=(pattern=1 color=red);
      enddiscreteattrmap;
```

```
        discreteattrvar attrvar=subject var=subject attrmap="subject";

        layout overlay/
            yaxisopts= (label="Serum kalium (mmol/L)" linearopts=(viewmax=6
            viewmin=2 TICKVALUELIST=(2 3 4 5 6)))
             xaxisopts= (label="Visit") ;
            seriesplot x=VISIT y=LBSTRESN/group=subject datalabel=LBSTRESC
            display=all name="a1";
            scatterplot x=VISIT y=LBSTRESN/group=subject;
            referenceline y=eval(coln(3.5, 5.5))/lineattrs=(pattern=dash
            color=red) curvelabel1=eval(colc("low", "high"));
            discretelegend "a1" / location=inside halign=right valign=top
            across=1;
        endlayout;

      endgraph;
     end;
 run;

 proc sgrender data=final template=scatter_series;
run;
```

For the same line plot output result, from the comparison of the code amount of the SGPLOT and GTL, it can be seen that the code of SGPLOT is less than that of GTL code.

From the statements used by the code, it can be seen that when defining the plot type, SGPLOT uses series and scatter statements, GTL uses seriesplot and scatterplot statements. In characterizing the X and Y axes, SGPLOT uses the XAxis/ YAxis statement and GTL uses the xaxisopts/ yaxisopts statement. When defining legend, SGPLOT uses keylegend, and GTL uses discretelegend. When defining the reference line statement, SGPLOT uses the refline statement, which can directly add the value of multiple lines, while GTL uses the referenceline statement, which needs the two functions COLN() and COLC() to define the value of multiple reference lines

In terms of the way defining the attributes, GTL adds a lot of options for the setting of attributes, which also makes our graphic codes more complex and requires more thought to adjust the graphics attributes. For example, for group attributes, we can see from the code that GTL uses discreteattrmap to define the attributes of different groups of the plot, and then uses the discreteattrvar statement to connect the discrete value attribute with the grouping variable and defines a new name similar to the role of the grouping variable. The statements for discreteattrmap and discreteattrvar are not in the layout overlay--endlayout area, but in the larger begingraph--endgraph area. And in the process of plotting, I find it that if the attribute of group is defined in discreteattrmap and the attribute is set in the following graph statement, the attribute set in the graph statement will override the previously set attribute, thus making the attribute of group defined in discreteattrmap invalid.

As for SGPLOT, in addition to defining a new data and using dattrmap=dataset to define the color and style of the group, you can use the STYLEATTRS statement which is part of the SGPLOT procedure beginning with SAS 9.4, to define group attributes directly. For example, the above code uses the DATACONTRASTCOLORS= option to specify the colors for the marker symbols. In addition, the DATASYMBOLS= option can specify the symbols that you want to use. it should be noted that, the attributes that are listed in the STYLEATTRS statement are associated with the group values in the order in which they appear in the data set .

Another interesting point I found is that as for legend statement. For GTL, there are two types of legends: discrete legend and continuous legend. For example, when you use seriesplot and scatterplot to make a plot, you generally need to use mergedlegend to merge the two legends. As shown in the code, I add the option display=all in the seriesplot statement, and then use discretelegend, it can still output the merged

legend. In addition, for SGPLOT, we can use the legenditem statement to define the legend attribute directly.

These are the differences in the code between SGPLOT and GTL. Simply looking at the amount of code, we can see that SGPLOT is simpler than GTL, and GTL has more options to choose, which also takes us more time. We can define the same result with simpler code. So for line charts, I recommend using SGPLOT.

2. SWIMMER PLOT

Scenario simulation: The statistician wants to see the response results to tumor assessments in six subjects and needed to mark each assessment with a different symbol and to mark the reason for the EOT with a different shape at the end of the swimmer plot, as shown in the figure 2.

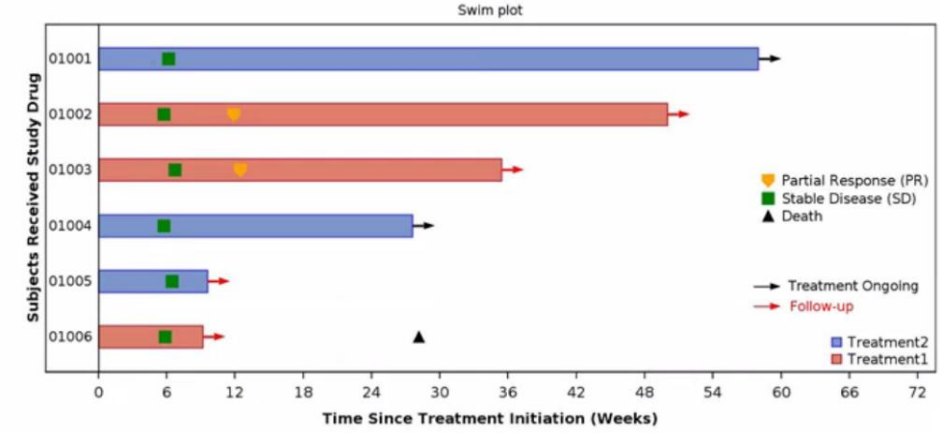

Figure 2. Swimmer plot

SGPLOT code:

```
proc sgplot data=_final1 nowall dattrmap=attrmap sganno=anno;
  title "Swim plot";
     hbarparm category=id response=xtime/group=trt attrid=trt barwidth=0.4
     name="trt" ;
     yaxis type=discrete display=(noticks novalues) label="Subjects
      Received Study Drug" labelattrs=(weight=bold size=9pt) ;

     xaxis type=linear label="Time Since Treatment Initiation (Weeks)"
     valueattrs=(size=8) labelattrs=(weight=bold size=9pt)  offsetmin=0
     values=(0 to 72 by 6);
     yaxistable idsub /location=inside position=left labelattrs=(size=9)
     NOLABEL;

     scatter x=prdy y=id/markerattrs=(color=Orange size=11
     symbol=homedownfilled ) name="pr" legendlabel="Partial Response (PR)" ;
     scatter x=sddy y=id/markerattrs=(color=Green size=11
     symbol=squarefilled ) name="sd" legendlabel="Stable Disease (SD)" ;
     scatter x=dthdy y=id/markerattrs=(color=black size=10
     symbol=trianglefilled ) name="dth" legendlabel="Death" ;
     scatter x=eosdy y=id/markerattrs=(color=magenta size=10
     symbol=DiamondFilled ) name="eos" legendlabel="End of Study" ;
     keylegend 'trt'/ noborder location=inside position=bottomright
     valueattrs=(weight=NORMAL size=9) across=1;
     keylegend 'pr' 'sd' 'dth' 'eos'/noborder location=inside
     position=right across=1 valueattrs=(size=9);
run;
```

GTL code:

```
proc template;
    define statgraph swimplot;
    begingraph;
    entrytitle "Swim plot";
        discreteattrmap name="stocks" / ignorecase=true ;
          value "Treatment1" / lineattrs=(color=white pattern=solid);
          value "Treatment2" / lineattrs=(color=white pattern=solid);
        enddiscreteattrmap ;
        discreteattrvar attrvar=stocks var=trt attrmap="stocks";
          layout overlay/
             xaxisopts=(label="Time Since Treatment Initiation (Weeks)"
             display=all labelattrs=(weight=bold) offsetmin=0.06
             offsetmax=0.02 linearopts=(tickvaluepriority=true
             tickvaluesequence=(start=0 end=72 increment=6)))
             yaxisopts=(label="Subjects Received Study Drug" reverse=true
             display=(label) linearopts=(tickvalueformat =sub.)
             labelattrs=(weight=bold));
             barchart category=idn response=xtime / group=trt
             orient=horizontal barwidth=0.4 name="stocks";
             axistable y=idn value=idsub /display=(values)
             valueattrs=(size=8pt weight=normal) valuejustify = left ;
             annotate/ID="12";

             scatterplot x=prdy y=idn/markerattrs=(color=Orange size=11
             symbol=homedownfilled ) name="pr" legendlabel="Partial
             Response (PR)" ;
             scatterplot x=sddy y=idn/markerattrs=(color=Green size=11
             symbol=squarefilled ) name="sd" legendlabel="Stable Disease
             (SD)" ;
             scatterplot x=dthdy y=idn/markerattrs=(color=black size=10
             symbol=trianglefilled ) name="dth" legendlabel="Death" ;
             scatterplot x=eosdy y=idn/markerattrs=(color=magenta size=10
             symbol=DiamondFilled ) name="eos" legendlabel="End of Study" ;
             discretelegend 'stocks'/location=inside border=false
             halign=right valign=bottom valueattrs=(weight=NORMAL
             family="Arial" size=9) down=1 ACROSS=1;
             discretelegend 'pr' 'sd' 'dth' 'eos'/location=inside
             border=false halign=right across=1 valueattrs=(size=9);
          endlayout;
      endgraph;
    end;
run;
```

From the two drawing codes, we can see that, for the definition of horizontal columns, SGPLOT uses the hbarparm statement to plot the data and overlay other scatter graphs on the bar chart. GTL uses the barchart statement, and uses the orient=horizontal option to make the posts horizontal. For the subject number on the left, SGPLOT directly uses the the yaxistable statement, and GTL uses axistable statement and additionally defines the Y-axis.

In addition, both legend and additional characters such as arrow and text use ANNOTATE Statement. In SGPLOT, the program directly defines an anno data set, and then directly defines it by sganno=anno at the beginning of PROC SGPLOT statement. In GTL, the program directly defines the id="12" variable and other attributes variables in the anno data set, and then defines annotate/ID="12" in the PROC template process.

For the convenience of less code and more simplicity, I recommend using SGPLOT to make swimmer plot.

3. Kaplan-Meier Curve

Scenario simulation: The statistician is about to have a project meeting and needs to review the KM survival curve for Progression-Free Survival in two dose groups. The remaining patients who do not have an end-point event are summarized in number at risk, as shown in the figure 3.

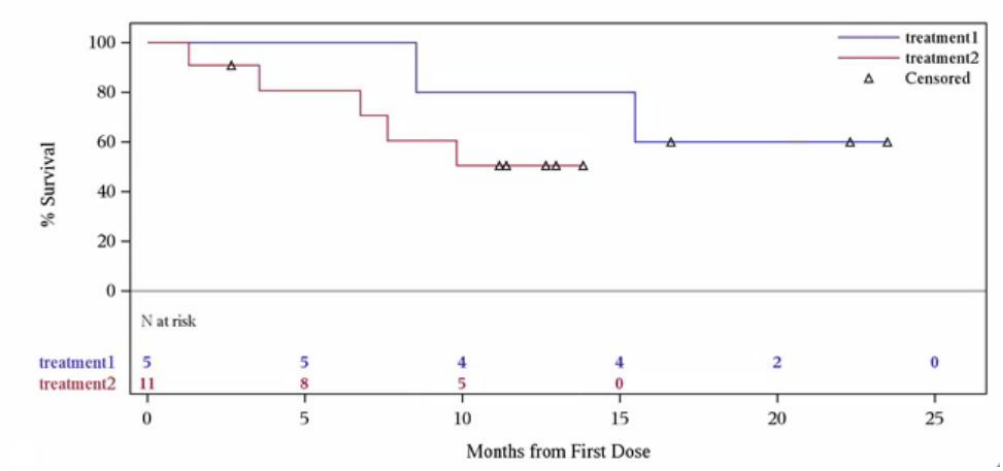

Figure 3. Kaplan-Meier Curve

SGPLOT code:

```
proc sgplot data=final;
   step x=time y=survival_ / group=STRATUMC lineattrs=(pattern=solid)
   name='s';
    scatter x=time y=censored_ / markerattrs=(symbol=triangle) name='c';
    scatter x=time y=censored_ / markerattrs=(symbol=triangle color=black)
    GROUP=STRATUMC;
     yaxis label="% Survival" values=(0 to 100 by 20) offsetmin=0.1
     offsetmax=0.1;
    xaxis label="Months from First Dose" offsetmin=0.02 offsetmax=0.02;
    refline 0;
    keylegend 's' 'c'/ location=inside position=topright across=1 noborder;

    xaxistable atrisk/x=tatrisk location=inside class=STRATUMC
    colorgroup=STRATUMC valueattrs=(size=9pt weight=bold);
    inset ("N at risk"="")/position=bottomleft;
run;
```

GTL code:

```
proc template;
    define statgraph kmplot;
    begingraph/border=false;
    entrytitle '';
    layout lattice / rows=1 rowgutter=1 ;
      layout overlay /
        xaxisopts=(label="Months from First Dose" offsetmin=0 offsetmax=0.06
        linearopts=(viewmin=0 viewmax=25 tickvaluesequence=(start=0 end=25
        increment=5) THRESHOLDMIN=0  THRESHOLDMAX=0) offsetmin=0.02
        offsetmax=0.06)
```

```
        yaxisopts=(label='% Survival' linearopts=(viewmin=0 viewmax=100)
        offsetmin=0.17 offsetmax=0.06);

        stepplot x=time y=survival_ / lineattrs=(pattern=solid
        thickness=1px) group=STRATUMC name='s' includemissinggroup=false;
        scatterplot x=time y=censored_ / name='c' includemissinggroup=false
        MARKERATTRS=(symbol=triangle color=black);
        discretelegend 's' 'c'/location=inside border=false halign=right
        valign=top down=1 ACROSS=1;
        annotate ;

        innermargin / align=bottom;
           AxisTable Value=atrisk X=tatrisk / Class=STRATUMC
           TITLEATTRS=GRAPHTITLETEXT valueattrs=(size=9pt weight=bold)
           colorGroup=STRATUMC Display=(Header Label) ;
        endinnermargin;
        referenceline y=0;
      endlayout;
    endlayout;
  endgraph;
  end;
run;
```

For KM curves, It is pretty obvious that SGPLOT has a lot less code than GTL. SGPLOT can define the X-axes and Y-axes with shorter codes. For example, SGPLOT can use the yaxis statement and directly define the scale of the Y-axis by using values = (0 to 100 by 20). while GTL needs to use the yaxisopts statement, and use the option linearopts = (viewmin=0 viewmax=100). In addition, for adding N at risk scales on the X-axis, SGPLOT directly uses xaxistable statement, which is just a single line of code, while GTL needs to insert axistable statement in a innermargin module.

It can be seen that for KM curves, it is wise to preferentially choose SGPLOT.

4. FOREST PLOT

Scenario simulation: The statistician is about to publish an article, which requires to combine Objective Response Rate results across multiple subgroups. It centers on a reference line, and describes the median and %95 CI of the results in each subgroup with multiple line segments parallel to the horizontal axis, as shown in the figure 4.

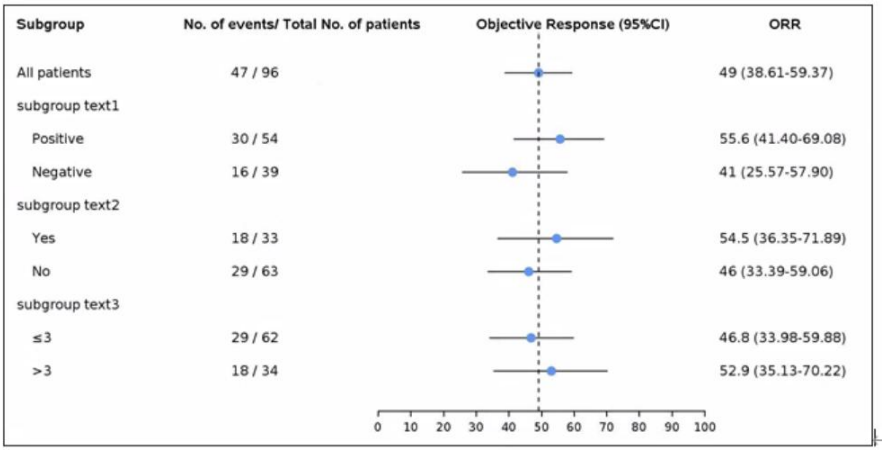

Figure 4. Forest plot

SGPLOT code:

```
proc sgplot data=mine dattrmap=attrmap sganno=anno;
    title "Forest plot" ;
    styleattrs axisextent=data;
    highlow y=obsid low=countnl high=countnh;
   scatter y=obsid x=countn / markerattrs=(symbol=CircleFilled
   color=cx6495ED size=10 ) ;
    scatter y=obsid x=countn / markerattrs=(size=0) x2axis ;
    refline 49.5 / axis=x lineattrs=(pattern=shortdash color=black);

   yaxistable subgroup  / location=inside position=left textgroup=id
    pad=(left=20px) labelattrs=(size=9 weight=bold) textgroupid=text
    indentweight=indentWt label='Subgroup' labeljustify=left;
   yaxistable countpct / location=inside position=left pad=(left=90px
   right=5px) labelattrs=(size=9 weight=bold ) labeljustify=left
   valuejustify=left LABELHALIGN=right valueattrs=(size=8);
   yaxistable orr / location=inside position=right pad=(right=20px)
   label="ORR" labelattrs=(size=9 weight=bold) valueattrs=(size=8);

   yaxis reverse display=none colorbands=odd
   colorbandsattrs=(transparency=1) offsetmin=0.1 offsetmax=0.1;
    xaxis display=(nolabel) values=(0 10 20 30 40 50 60 70 80 90 100) ;

   x2axis label='Objective Response (95%CI)' display=(noline noticks
   novalues) labelattrs=(size=9 weight=bold);
run;
```

GTL code:

```
proc template;
    define statgraph Forest_Plot;
    begingraph;
    entrytitle "Forest plot";
       layout lattice / columns=4 columngutter=0 columnweights=(0.25 0.17
       0.4 0.17) rows=1 opaque=true;
        sidebar / align=top;
          layout lattice / rows=1 columns=4 columnweights=(0.17 0.3 0.3
              0.17) opaque=true;
             entry textattrs=(size=9 weight=bold) halign=left "Subgroup";
              entry textattrs=(size=9 weight=bold) "No. of events/ Total No.
              of patients";
              entry textattrs=(size=9 weight=bold) "Objective Response
             (95%%CI)";
              entry textattrs=(size=9 weight=bold) "ORR";
          endlayout;
       endsidebar;

       layout overlay /
          xaxisopts=(display=none offsetmin=0.0 offsetmax=0.0)
           axisopts=(reverse=true display=none tickvalueattrs=(weight=bold)
           offsetmin=0.1 offsetmax=0.1) walldisplay=none;
           axistable y=obsid value=subgroup /indentWeight=indentwt
           display=(values) valueattrs=(family="Arial" size=8pt
           weight=normal) valuejustify = left;
       endlayout;

       layout overlay /
          xaxisopts=(display=none offsetmin=0.0 offsetmax=0.0)
```

```
            yaxisopts=(reverse=true display=none ) walldisplay=none;
            axistable y=obsid value=countpct /display=(values)
            valueattrs=(family="Arial" size=8pt weight=normal) valuejustify =
            left ;
        endlayout;

        layout overlay /
            xaxisopts=(label=" " linearopts=(tickvaluepriority=true
            tickvaluelist=(0 10 20 30 40 50 60 70 80 90 100)) display=(line
            ticks tickvalues) tickvalueattrs=(size=7pt) offsetmin=0.0
            offsetmax=0.0)
            yaxisopts=(reverse=true display=none) walldisplay=none;
            scatterplot y=obsid x=countn / xerrorlower=countnl
            xerrorupper=countnh markerattrs=(symbol=CircleFilled
            color=cx6495ED size=10) errorBarCapShape=none
            DATALABELSPLITJUSTIFY=left;
            referenceline x=49.5/lineattrs=(pattern=shortdash color=black);
        endlayout;

        layout overlay /
            xaxisopts=(display=none offsetmin=0.0 offsetmax=0.0)
            yaxisopts=(reverse=true display=none) walldisplay=none;
            axistable y=obsid value=orr /display=(values) valueattrs=(size=8pt
            weight=normal) valuejustify = left ;
        endlayout;

    endlayout;
    endgraph;
    end;
run;
```

For the forest plot, the intuitive feeling is that the amount of code for both is more.

SGPLOT uses the yaxistable statement to distribute the content on the left and right sides, and uses a HIGHLOW statement and scatter statement to plot the upper and lower limits of the 95% CI and the median of the Objective Response Rate. The column " obsid " is used to position the segments vertically, the segments are drawn along the x-axis using the various variables such as Low, High. The default option here is Type = line. It's worth mentioning that you can also use HIGHLOW statement when drawing a swimmer plot, and draw the columns by defining type=bar, and the rest of the text needs to be listed horizontally.

For GTL, it uses scatterplot statement and xerrorlower-xerrorupper option to define the 95%CI, and defines the marker of the median using the option of markerattrs. The rest of the text content uses axistable statement in multiple layout overlays.

One point worth mentioning is that the adjustment of the header content. At the beginning, I use SGPLOT to make the forest plot by using yaxistable statement to directly define the label of the header. However, because the text description “No. of events/ Total No. of patients” is too long, the running results would automatically wrap, resulting in unsightly output, as shown in Figure 5. I try to use the pad option in yaxistable to adjust the distribution value of these four contents, but the result just can only adjust the distribution size of the contents below the table head, not the width of the table head text. Finally, by defining anno separately, increasing the width value, I successfully get the table head without automatic newline. Unfortunately, the four-section header also looks less neat than the GTL output. GTL can output neat results as above by directly defining four parts through the layout lattice in sidebar, and the position of the header content can be adjusted at will.

Figure 5. Output header- unsightly version

For those reasons, although SGPLOT can also complete the forest plot, I recommend GTL to be on the safe side.

5. Specific comparison of the two plot methods

characteristics	GTL	SGPLOT
Title	entrytitle	title
Reference lines	referenceline	refline
Output	proc sgrender data	×(direct output)
Controls the axes	xaxisopts/yaxisopts	xaxis/yaxis
Type statement	scatterplot/ stepplot	scatter/step
Annotation	annotate;	sganno=anno;
Legend	discretelegend/ mergedlegend	keylegend
Group attributes	discreteattrmap	dattrmap

Table 1. GTL and SGPLOT code difference

CONCLUSION

Among the various statistical graphics requirements, such as simple line plot, swimmer plot, KM plot, it is recommended to use SGPLOT, which can make your program code less. For more complex graphs such as forest plot, GTL can achieve better results. SGPLOT statements are more direct and have fewer options, while GTL has more options, which can adjust the graph through more detailed options, but also increase the learning time and difficulty to programmers. In SAS 9.3 and later version, SGPLOT adds more features and performance to improve plotting, which allows users to further customize their graphs. In brief, for simple single-cell graphs, SGPLOT is a smart choice. Of course, there is no doubt that GTL is invincible.

REFERENCES

Matange, Sanjay. March 2016. Clinical Graphs Using SAS®. Cary, NC, USA: SAS Institute Inc.

Lelia McConnell. “Assigning graph style attributes easily! ”. May 30, 2014. https://blogs.sas.com/content/sgf/2014/05/30/assigning-graph-style-attributes-easily/

ACKNOWLEDGMENTS

I would like to thank my leader Chen Ju for the guidance and my colleague Guo Danting for the guidance and clarification.

CONTACT INFORMATION

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

< Yan He >
< Fosun Pharma >
< 13658070023>
< heyan1@fosunpharma.com >