PharmaSUG 2019 - Paper AD-111

Learning SAS® GTL Basics with Practical Examples

Jinit Mistry, Seattle Genetics Inc.

ABSTRACT

Over time, the business needs for data visualization have evolved due to the increased complexity of clinical trials. There has been an increased demand for complex graphs as part of clinical study reports, based on statistical analysis plans. For efficacy endpoints of oncology trials, complex graphs such as KM plots, forest plots, waterfall plots, spider plots, and box plots require creating more robust and customizable SAS® code. To address this need, various statistical graphics (SG) procedures and Graph Template Language (GTL) have been introduced in SAS®. SAS® SG procedures use an associated GTL template in the background because SG procedures have limited ability to create graphs. This makes GTL critical to understand in order to create customizable templates and associate them with the data to accomplish the goals of the analyses. GTL can create more customizable graphs over SG procedures to address such complex graphical needs. This paper talks about GTL key statements and how GTL uses a template that can work for different scenarios. Some relevant industry examples will also be shared in this paper.

INTRODUCTION

SAS® ODS graphics system has introduced Graph Template Language (GTL) with the SAS® 9.2 release. GTL is powerful and very rich in features but this flexibility and power come with some complexity. Statistical procedures generate an aesthetically pleasing graph by default using the following statement:

```
ODS Graphics ON;
```

Often times SAS® users struggle to customize graphs due to lack of GTL and PROC TEMPLATE knowledge. PROC TEMPLATE has been extended to define a new type of template called STATGRAPH with GTL syntax. The biggest challenge SAS® users face with understanding PROC TEMPLATE and STATGRAPH is realizing that it's not like a traditional SAS® PROC MEANS or FREQ that has a set of statements and options to describe its behavior, instead it is a very flexible way of defining a generalized template to address specific goals.

In fact, there are various statistical procedures such as SGPLOT, SGPANEL, and SGSCATTER that behave similarly to traditional SAS® procedures. The SG procedures have an associated specific GTL template that is working in the background. Most of the graphical needs can be addressed using SG procedures, but in certain situations there may be a need for more customized graphs that are beyond capabilities of SG procedures. Thus understanding GTL is very important to make more customizable graphs and meet the needs of graph reviewers.

This paper intends to motivate programmers to learn GTL by introducing basics of GTL, GTL statements based on categories, and the ODS family of GTL, as well as touching on the development process for complex graphs such as the Kaplan-Meier (KM) plot using GTL. A reference table of key statements, their functionality and respective application in statistical graph is included for programmer's guidance.

GTL BASICS AND GTL COMPONENTS

GTL is an extension to the Output Delivery System (ODS) to support statistical graphics. GTL syntax is defined by the PROC TEMPLATE procedure. This special template to define graph components is called STATGRAPH template. All STATGRAPH template definitions must start with a BEGINGRAPH statement and end with an ENDGRAPH statement. GTL statements can be used inside the BEGINGRAPH block. The PROC TEMPLATE typical syntax is shown below to define STATGRAPH:

```
PROC TEMPLATE;
  DEFINE STATGRAPH graph-path </STORE=libref.template-store>;
   DYNAMIC variable-1<'text-1'><variable-n<'text-n'>>;
    MVAR variable-1<'text-1'><variable-n<'text-n'>>;
   NMVAR variable-1<'text-1'><variable-n<'text-n'>>;
   NOTES 'text';
     BEGINGRAPH </option(s)>;
         <GTL-global-statements>
           GTL-layout-block
         <GTL-global-statements>
      ENDGRAPH;
    END;
   RUN;
```

GTL has structured syntax which provides a building-block approach. Executing PROC TEMPLATE compiles and saves the template; executing PROC SGRENDER with data= and template= options associates data with the template and creates the actual graph at run time. Note that specifying a template that is not a STATGRAPH template is not supported. If the programmer specifies a non-STATGRAPH template, then the SGRENDER procedure might produce unpredictable results. The typical syntax to associate STATGRAPH with a data set is given below:

```
PROC SGRENDER TEMPLATE=<statgraph-template-name> DATA=<input-dataset-name>;
  <other optional statements>
Run;
```

In order to create any graph, the programmer needs to understand desired graph elements and then relate GTL components with those desired graph elements. With reference to any visual graph elements, key statements of GTL can be divided into four categories (Matange, "GTL Introduction", 2008) as follows:

1. Plot statements: The significance of the plot statements is to determine how data is depicted. GTL supports a variety of plot category statements such as basic plots (STEPPLOT, SCATTERPLOT, SERIESPLOT, etc.), categorical plots (BARCHART, PIECHART, etc.), distribution plots (BOXPLOT, DENSITYPLOT, HISTOGRAM, etc.), fit plots (REGRESSIONPLOT, LOESSPLOT, etc.), and many more. One important feature of GTL is that multiple plot statements can be combined to create complex graphs.
2. Layout statements: The significance of layout statements is to determine where the plots are drawn. GTL supports a variety of containers or layouts based on requirements. Layouts are used to manage the graph area, creating single-cell or multi-cell graphs. Layouts can be nested inside other layouts to create complex graphs but note that there can be only one root layout statement inside the BEGINGRAPH block. Some of the important LAYOUT statements are as follows:
    - LAYOUT OVERLAY: The LAYOUT OVERLAY statement builds a 2-D, single-cell graph by overlaying the results of the statements that are contained in the LAYOUT block. It is very flexible and hence considered the most useful single-cell container. Here all 2-D plot statements are used except SCATTERPLOTMATRIX statement.
    - LAYOUT LATTICE: The LAYOUT LATTICE is most useful and flexible multi-cell container for an adhoc arrangement of cells. It can have different plot types. The axes can be independent or shared. Each cell can have nested layouts, enabling the programmer to create almost any arrangement.
    - LAYOUT GRIDDED: The LAYOUT GRIDDED provides automatic alignment of respective grid cells. it is similar to LAYOUT LATTICE except cells have to be equal in gridded layout and only HTML and PRINTER ODS destinations are supported. Mostly used for creating

tables of statistics embedded in a graph.

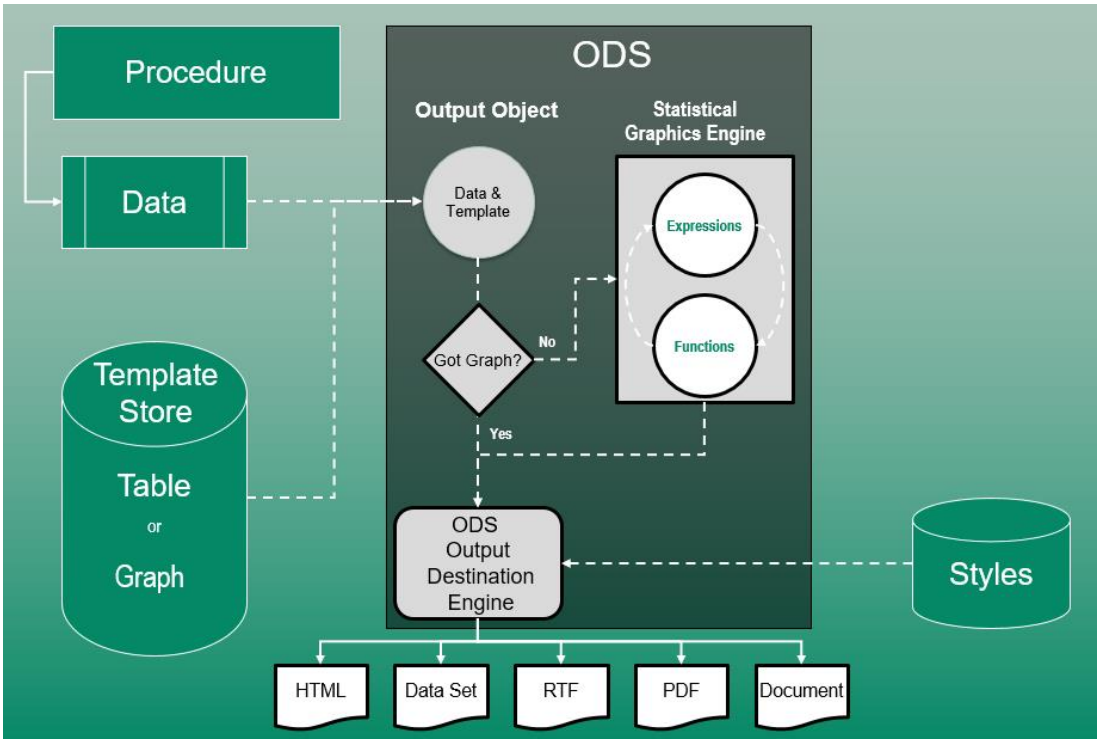
- LAYOUT DATALATTICE: The LAYOUT DATALATTICE is used to create gridded panels by classification variables and supports up to two class variables, one for a row variable and one for a column variable. In a LAYOUT DATALATTICE statement, the programmer can specify the row variable in the ROWVAR= option and column variable in the COLUMNVAR= option.
- LAYOUT DATAPANEL: The LAYOUT DATAPANEL supports a list of class variables. The number of rows and columns are controlled by statement options. In the LAYOUT DATAPANEL statement, the programmer can specify the classification variables with CLASSVARS= option. There are other options available to control various aspects of the panel. PROTOTYPE block is used to define the contents for each cell.
- LAYOUT PROTOTYPE: The LAYOUT PROTOTYPE can be used in DATAPANEL and DATALATTICE classification panels. It can have any combination of compatible basic plot statements. However, it cannot contain distribution or fit plots.

3. Accessory statements: There are various accessory statements such as titles, footnotes, entries, axes, legends, attribute maps, and styles. They provide the programmer good control over enhancements of graph components. For example, legends support the reviewers' interpretation of the graph. Textual information provided by titles/footnotes makes the graph more legible.
4. Conditionals, expressions, functions and more: GTL provides additional capabilities to support conditional statements, expressions, macros, DYNAMIC, etc. these features help GTL make a more flexible and robust template. For example, dynamically defined variables are used as parameter values that are evaluated at the time of graph rendering, and not compile time. It helps create a graph template that is more flexible at execution time.

Please refer to References (Matange, "GTL book", 2013) and (Matange, "GTL Introduction", 2008) for detailed information. Once programmers understand key components of GTL, it enables them to mix and match key statements in existing templates and modify them to achieve the desired graph.

ODS FAMILY OF GTL OVERVIEW

To get more control over statistical graphs, it is important to have a high-level understanding of the ODS family working model in the background. Graphs that are produced by ODS Graphics are controlled by options, the data object (the matrix of information that is graphed), a style template, and a graph template. A style template is a SAS program that controls the overall appearance of graphs, including colors, line and marker styles, sizes, fonts, and so on. A graph template is a SAS program written in the GTL that provides a detailed specification of the layout and contents of each graph. Below in Display 1, the general work flow model of the ODS family is shown.



Display 1. ODS Family General Work Flow Model

Display 1 shows ODS family members and how they work together to produce final output. ODS family members include procedures which generate data objects, a template store that has a collection of compiled templates for table or graph, an output object that is a combination of data and template objects, a statistical graphics engine that processes expressions and functions to make more flexible templates, and style templates associated with the active ODS destination.

Template Store

SAS stores certain types of information in data sets called item stores. Templates that PROC TEMPLATE has compiled are stored in item stores called template stores. There are primarily two template stores for any table or graph. It can be seen by statement "`ods path show;`" It will show sasuser.templat(Update) and sashelp.tmplmst(Read).If the programmer modifies any existing template, it is stored in sasuser.templat. Note that options specified in brackets ('Update' and 'Read') are default permissions for the SAS user for the template store. Template store sashelp.tmplmst must not be modified, as modifications to this template store can corrupt the SAS system. That is the reason why by default SAS stores modified templates in sasuser.templat.

The programmer can get source code of the template store for the existing statistical graphs produced by ODS graphics. There is a template name associated with the output object. It can be found in the log by the '`ODS TRACE ON;`' statement. Later the source statement with template name associated with output object is used to get STATGRAPH code. For example, if the programmer wants to find out the default STATGRAPH template used for a KM plot then use the '`ODS TRACE ON;`' statement before the PROC LIFETEST statement. It should show the STATGRAPH template name in the log file, which is Stat.Lifetest.Graphics.ProductLimitSurvival. Please see code below to view the source code:

```
PROC TEMPLATE;
SOURCE Stat.Lifetest.Graphics.ProductLimitSurvival;
Run;
```

Note that there is a total of 2 default STAGRAPH templates available for KM plots. The default STATGRAPH template is shown in the above example, which provides an at-risk table inside the body of

the graph. The second STATGRAPH template stat.Lifetest.Graphics.ProductLimitSurvival2 provides an at-risk table outside the body of the graph.

ODS Styles

An ODS style definition is composed of a set of style elements. A style element is a collection of style attributes that applies to a particular feature or aspect of the output. A value is specified for each attribute in a style definition. For example, GraphFit is the style element used for fit lines. GraphFit attributes include LineThickness, LineStyle, MarkerSize, MarkerSymbol, ContrastColor, and Color.

ODS Graphics is extensively using ODS Styles. All graphical features of GTL derive their default visual attributes from specific style elements. For example, the Axis Tick value graph element uses the 'GraphValueText' style element. Basic plot statements like scatter, series, step, histogram, etc, use the style element 'GraphDataDefault' when ungrouped and 'GraphData1'-'GraphData12' when grouped. For an important list of GTL graphical elements, refer to (Matange, "GTL Introduction", 2008). A complete list of style elements affecting template based graphics is given in SAS 9.2 Product documentation for the Output Delivery System. It is also mentioned in the recommended reading section.

EXAMPLE OF KM PLOT FOR GTL LEARNING

In the oncology therapeutic area, analysis of survival data is often done by measuring endpoints such as overall survival, progression-free survival, etc. The Kaplan-Meier or Product Limit estimate plot is the preferred visual method for analyzing such survival endpoints. Since it is a very popular plot in survival analysis, the KM plot has many features to change and enhance through procedure options, graph templates, and style templates.

- To begin with, the programmer has to analyze the requirements of the desired KM plot. For instance, consider a programmer who needs to create a KM plot for overall survival in the ITT population. Other requirements are to show an embedded statistics table containing p-value, hazard ratio, 95% CI and number of events per treatment inside the KM plot. To add even more, show the number of patients at risk in the form of a table outside the KM plot in 6-month intervals starting from 0 Month to 54 Months. Also, the programmer needs to insert a legend to show color-coded lines for different treatment group values. Visual representation of the desired graph is given in Figure 1 for a better understanding of the graph specification.

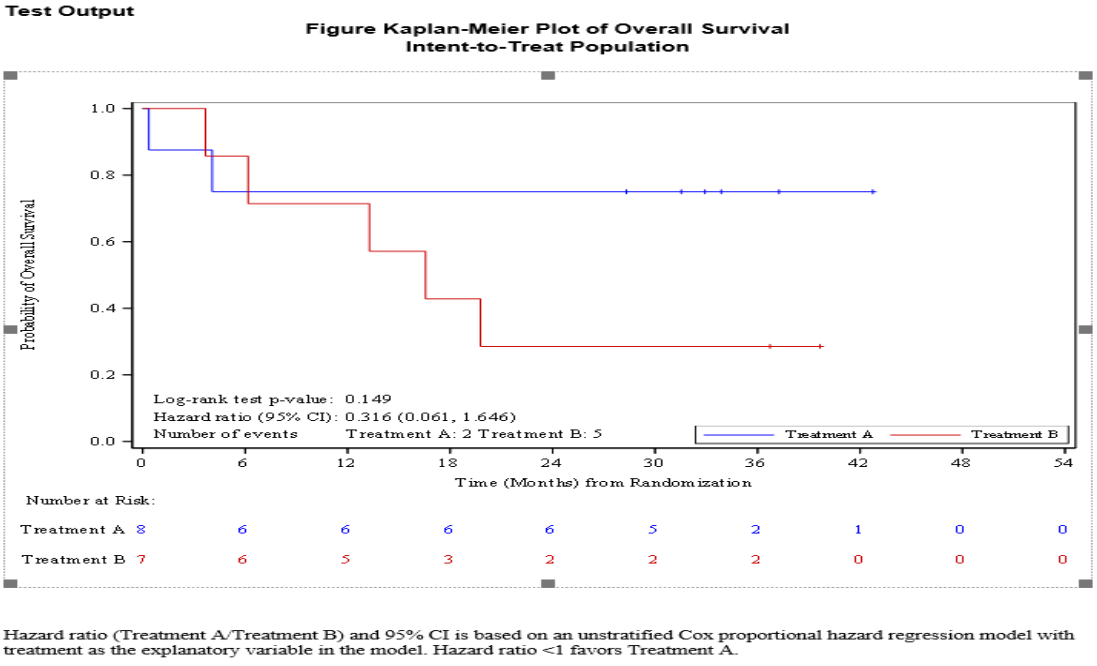


Figure 1. Desired KM Plot

- The next step is to understand input data set variables and processing of that data set if needed. It is assumed that ADTTE is the input data set to generate this KM plot. The required variables are AVAL which contains time to event values and CNSR which contains censor and event information. The optional variable is for stratification which should be the treatment variable. A dummy ADTTE dataset for demonstration purposes is shown in Display 2. Note that CNSR=1 indicates censored subjects.

	USUBJID	TRT01PN	AVAL	CNSR
1	TEST-X14	2	39.655030801	1
2	TEST-X3	1	37.256673511	1
3	TEST-X15	2	36.73100616	1
4	TEST-X4	1	33.872689938	1
5	TEST-X5	1	31.540041068	1
6	TEST-X6	1	28.320328542	1
7	TEST-X7	1	42.743326489	1
8	TEST-X8	1	32.919917864	1
9	TEST-X9	2	3.6796714579	0
10	TEST-X1	1	4.0739219713	0
11	TEST-X2	1	0.3613963039	0
12	TEST-X10	2	13.273100616	0
13	TEST-X11	2	16.558521561	0
14	TEST-X12	2	19.778234086	0
15	TEST-X13	2	6.1765913758	0

Display 2. Test ADTTE Dataset

- The KM plot is generated using the PROC LIFETEST procedure, which also has many options for customizations. A Kaplan-Meier figure can be produced with the LIFETEST procedure with the PLOTS=survival option by enabling the '`ODS graphics on;`' statement. The Plots option also provides various options to control the at-risk table. The default graph generated from the code is shown in figure 2. Note that by default the at-risk table is shown inside the axis. Here the Atrisk option is used to get the patients-at-risk table. Below a sample syntax of PROC LIFETEST is shown:

```
PROC LIFETEST data=adtte method=pl plots=survival(atrisk=0 to 54 by
6)          outsurv=survival ;
     STRATA trt01pn/test=logrank;
     TIME aval*cnsr(1);
run;
```

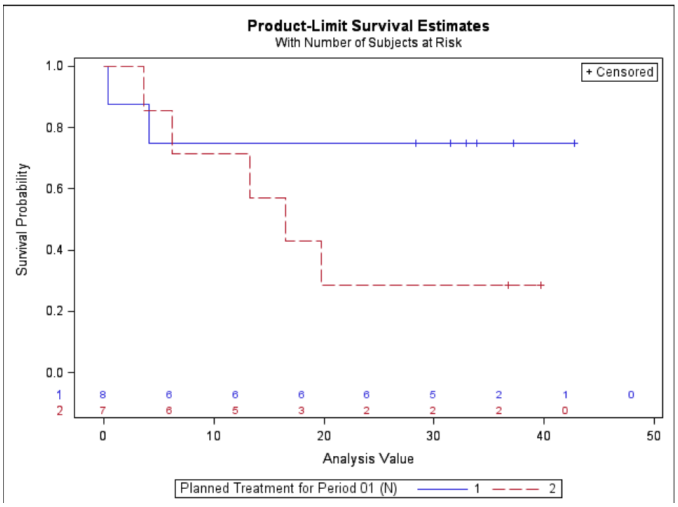


Figure 2. Default KM Plot of PROC LIFETEST With (Atrisk=0 To 54 By 6) Option

- Based on figure 2, data shown on the x-axis is through 50 months. Here by default it has an interval of 10 months on the x-axis. Note that it is missing the 6-month interval and does not have the upper limit as 54 months. This is happening by default based on availability of data. The programmer can process data to get 54 months. Based on a statistical algorithm, the programmer can get summary statistics to display in graph as per requirement. For example, the hazard ratio and 95% CI can be derived from the PROC PHREG model and '`ODS output hazardratios=hr;`' statement. The objective of this paper is to cover the GTL topic and assumes that the programmer has done data processing as required for the graph.
- After analyzing desired graph details, setting up the necessary input dataset and macro variables, the next step is to connect graph elements to GTL statements. As mentioned earlier in requirements of the desired graph, the programmer need to show an embedded statistics table, the KM plot and an at-risk table. For multi-cell graphs, a flexible container is LAYOUT LATTICE so the programmer can choose to use the root LAYOUT LATTICE statement. LAYOUT LATTICE is a flexible way of splitting the graph area. The programmer can use different combinations of other LAYOUT statements inside the LAYOUT LATTICE block.
- The KM plot can be managed in the graph area using the LAYOUT OVERLAY statement. The programmer can use the STEPPLOT statement with options x=time and y=response with group=stratum variable that draws STEPPLOT. Also a SCATTERPLOT statement with x=time and y=censored with group=stratum variable can be added to draw censored values. So the programmer can get a KM plot by combining STEPPLOT and SCATTERPLOT with necessary options. The legend can be customized by the DISCRETELEGEND statement. Since censored values are drawn by the STEPPLOT statement, the programmer can use the name='km' option in the STEPPLOT statement. In the DISCRETELEGEND statement, further visual attributes can be mentioned using available properties. For example, location=inside should be mentioned to show the legend inside the KM plot graph area.
- The LAYOUT GRIDDED container is frequently used to show embedded statistics. As per the requirement, a statistics table is needed inside the KM plot so the programmer can put the LAYOUT GRIDDED block inside of the LAYOUT OVERLAY block. The programmer can use valign=, halign= to control vertical and horizontal alignments and columns=2 option to specify number of columns in the table. GTL provides a variety of options and statements to control graph elements. To add a statistics table and display textual information, the programmer can use entry statements. Note that statistics values are created during the data preparation step and stored in macro variables. In GTL the programmer can use the MVAR statement to define macro

variables before BEGINGRAPH block. After defining mvars, the programmer can directly use defined macro variables in entry statements.

- For the last remaining part - the risk table - the programmer can add LAYOUT OVERLAY block nested to root LAYOUT LATTICE block, which should be outside of the LAYOUT OVERLAY block previously used for KM plot. This way programmer can ensure to create at-risk table outside KM plot area. Now for the risk table textual data, the axis needs to be aligned with the graph. The programmer can use the BLOCKPLOT statement with options x=time block=left1 to display the risk table. Note that left1 and time are variables previously derived during data preparation. left1 contains Treatment A patients-at-risk values. Similarly, the programmer can specify other block statements to show patients at risk for Treatment B. Based on selecting key statements as discussed above and with selection of the appropriate plot, layout, and visual attribute options, the desired graph is produced by customizing the style and graph templates. While associating template km_newstat1 with data by PROC SGRENDER, the survival plot shown in Figure 1 can be generated.

Below is the complete PROC TEMPLATE code for the programmer's reference:

```
PROC TEMPLATE;
  DEFINE style sty_km;
  parent=rtftnr10;
    STYLE graphfonts from graphfonts /
      'GraphDataFont'=("times new roman",10pt)
      'GraphDataValue'=("times new roman",10pt);
    STYLE body from body /
      leftmargin = 1.0in
      rightmargin = 1.0in
      topmargin = 1.0in
      bottommargin = 1.0in;

    STYLE graphdata1 from graphdata1 /color = blue contrastcolor = blue
linestyle=1; **Dark blue**;
    STYLE graphdata2 from graphdata2 /color = cxcc0000 contrastcolor =
cxcc0000 linestyle=1; **Dark red**;
    STYLE graphdata3 from graphdata5 /markersymbol = "plus"  contrastcolor =
blue;
    STYLE graphdata4 from graphdata6 /markersymbol = "plus"  contrastcolor =
cxcc0000;
  end;
run;

PROC TEMPLATE;
  DEFINE STATGRAPH km_newstat1;
      MVAR logrank hr eve;
      BEGINGRAPH / designwidth=9in designheight=7in border=false;
       ENTRYTITLE ' ';

        LAYOUT LATTICE /columndatarange=union rowweights=(0.82 .05 .13);

         /*** Survival Plot ***/
         LAYOUT OVERLAY / yaxisopts=(label= 'Probability of Overall Survival'

            labelattrs=(family="times new roman" size=9pt)
            tickvalueattrs=(family="times new roman" size=9pt)
            linearopts=(viewmin=0.0 viewmax=1 tickvaluesequence=(start=0.0
            end=1 increment=0.2)) cycleattrs=true
                  xaxisopts=(label="Time (Months) from Randomization"
                  labelattrs=(family="times new roman" size=9pt)
```

```
                tickvalueattrs=(family="times new roman" size=9pt)
                offsetmin=0.01 offsetmax=0.01
                linearopts=(tickvaluesequence=(start=0 end=54 increment=6)
                viewmin=0 viewmax=54 ));

                STEPPLOT x=time y=response/group=stratum name='km'
                connectorder=xaxis;

                SCATTERPLOT x=time y=censored/group=stratum name='tc';
                DISCRETELEGEND 'km' /location=inside
                autoalign=(bottomright) border=true
                valueattrs=(family="times new roman" size=9pt);

         /**** Statistics Embedded table in survival plot ****/

         LAYOUT GRIDDED /valign=bottom halign=left border=false pad=(left=10
         bottom=0) opaque=false columns=2 order=rowmajor;
           ENTRY halign=left "Log-rank test p-value:"  /
           textattrs=(family="times new roman" size=9pt color=black);
           ENTRY halign=left logrank   / textattrs=(family="times new roman"
           size=9pt color=black);
           ENTRY halign=left "Hazard ratio (95% CI):"  /
           textattrs=(family="times new roman" size=9pt color=black);
           ENTRY halign=left hr   / textattrs=(family="times new roman"
           size=9pt color=black);
           ENTRY halign=left "Number of events"  / textattrs=(family="times
           new roman" size=9pt color=black);
           ENTRY halign=left eve    / textattrs=(family="times new roman"
           size=9pt color=black);

          ENDLAYOUT; * of layout gridded;
         ENDLAYOUT; * of layout overlay;

        /***Subjects at risk ***/
          LAYOUT OVERLAY;
            ENTRY halign=left 'Number at Risk:' / pad=(left=10)
            textattrs=(family="times new roman" size=9pt);
          ENDLAYOUT;

          LAYOUT OVERLAY / walldisplay=(fill) xaxisopts=(display=none);

            BLOCKPLOT x=time block=left1 /display=(values label)
            repeatedvalues=true labelattrs=(family="times new roman"
            size=9pt) class=stratum
            valueattrs=(family="times new roman" size=9pt color=blue)
            valuehalign=start;

            BLOCKPLOT x=time block=left2 /display=(values label)
                    repeatedvalues=true labelattrs=(family="times new roman"
                    size=9pt) class=stratum
                valueattrs=(family="times new roman" size=9pt color=cxcc0000)
                valuehalign=start;
            ENDLAYOUT; * of layout overlay;
          ENDLAYOUT; * of root layout lattice at the very top;
    ENDGRAPH; * of begingraph block;
  END; * of Define STATGRAPH block;
run; * of PROC TEMPLATE;
```

GTL KEY STATEMENTS AND FREQUENTLY DEMANDED STATISTICAL GRAPHS

Following is a table of key statements that can be helpful for a programmer to develop survival graphs using GTL. Please check SAS® documentation for detailed information about available options for each statement to produce the desired graph. Note that LAYOUT-related statements are not included here since they are used to manage the graph area and tell the programmer where the graph is drawn. The LAYOUT statements are explained in SAS® documentation and other available online papers.

Key Statements	Functionality	Statistical Graph
ENTRY	An ENTRY statement creates one line of text in the plot area. The statement must be specified within a LAYOUT, HEADER, SIDEBAR, or CELL statement block. It cannot be specified outside of one of these blocks, where global statements like ENTRYTITLE and ENTRYFOOTNOTE are used.	Any Statistical Graph where textual information needs to be shown.
SCATTERPLOT	Creates scatterplot of input data. Forest plot can be produced with x= column variable or expression (e.g. can be hazard ratio variable), y= column variable or expression (e.g. can be textual discrete values) with plot options xerrorlower=HRLOWERCL xerrorupper=HRUPPERCL and other necessary visual attribute options. KM plot censored values can be drawn using options x=time y=censored as seen in earlier example.	Forest plot (E.g. various analysis to show hazard ratio, 95% CI in plot), KM Plot (e.g. To show censored values in OS, PFS, etc.), to show Mean score with 95% CI over time in various analyses.
STEPPLOT	Displays a series of horizontal and vertical line segments that connect observations of input data. KM plot can be produced with x=column variable or expression (e.g. time variable), y=column variable or expression (response variable). Additional plot statement for group=stratum can be added. In the example above, the connectorder=xaxis option was used which specifies how to connect the data points to form the step line. There are plenty of options available based on the need of the graph.	KM plot (e.g. OS, PFS, etc.).
SERIESPLOT	Displays a series of line segments that connect observations of input data. Spider plots can be produced with x=column variable or expression (e.g. duration since baseline), y=column variable or expression (e.g. tumor size relative to baseline) with required option as per need of the output.	Spider plot (e.g. to show relative change in tumor size with respect to time and baseline value).
BARCHART	Creates a bar chart computed from input data. Swimmer plots for AE events vs percentage of patients can be produced with x= column variable or expression (e.g. adverse event term), y= column variable or expression (e.g. percentage of patients) and orient= horizontal option. Similarly, waterfall plots can be generated by assigning appropriate variables in x=, y= and other required options as per need of the output.	Swimmer plot (e.g. certain events like AE vs percentage of patients), waterfall plot (e.g. change in size of tumor for each patient).

Key Statements	Functionality	Statistical Graph
DISCRETELEGEND	Displays the symbols, line patterns or colors for different levels of grouped data. It basically helps interpretation of plots when there is grouped data. Programmers need to specify the name= option along with the group= option in any plot statement and later reference it in the discretelegend statement with necessary options as per need of the output. Note that legends are not automatically displayed for plots with group values.	Any Statistical Graph where there is grouped data.
DISCRETEATTRMAP block and VALUE statements	Define discreteattrmap with particular name to assign discrete values. Values can be assigned by value statement with specifying fillattrs=, markerattrs= and lineattrs= options. For example, "Placebo" / fillattrs=(color=green) markerattrs=(color=darkgreen) lineattrs=(color=darkgreen); Each discrete value should have a separate value statement. Discrete blocks end by the enddiscreteattrmap statement.	Any Statistical Graph where we need to associate particular color property with discrete values of data.
DISCRETEATTRVAR	Define discrete attribute variables specifying attrvar= , var= and attrmap= options. Here it creates attribute map variables using the attrvar= option to associate the attribute map named by the attrmap= option with a needed variable set by the var= option. For example, treatment group variable values can be associated with the needed attribute map as mentioned. Note that usually different group value attributes are rendered using GRAPHDATA1 to GRAPHDATA12 elements based on the active style. It is assigned per the order of the input data, so there is no way it is assigning attributes consistently based on value. Attribute map statements are used to overcome this issue.	Any Statistical Graph where we need to associate particular color properties with discrete values of data.
DYNAMIC	Define dynamic variable(s). Here values of DYNAMIC-defined variables can be supplied by DYNAMIC statement value assignment in the PROC SGRENDER procedure.	Any Statistical Graph where we want to make the template flexible.
MVAR	Define macro variable(s). Here values can be supplied by either %Let statement or CALL Symput() function.	Any Statistical Graph where we want to make the template flexible.
NMVAR	Define macro variable(s) that resolve to a number(s). Here values can be supplied by either %Let statement or CALL Symput() function.	Any Statistical Graph where we want to make the template flexible.

Table 1. Key Statements, Their Functionality and Respective Application in Typical Statistical Graphs

CONCLUSION

Learning and understanding GTL basics and graph element-related ODS components gives the programmer power to create and customize simple graphs as well as complex, sophisticated graphs. In this paper we saw how to access and produce default statistical graphs with help of ODS and PROC TEMPLATE statements, which are handy to follow. By example of a KM plot approach, the programmer can cultivate an understanding of the development of graphs that require customization. Finally, key statements based on frequently requested graphs are shown to give GTL programmers a reference on development of corresponding statistical graphs. Since graphs provide data visualization and great insights about data to support quick interpretation for reviewers and GTL is a single system solution to all graph problems, it is worth investing time learning GTL to be able to handle more customized statistical graphs.

REFERENCES

Matange, S. (2008). "Introduction to the Graph Template Language". *Proceedings of SAS Presentations*

at SAS Global Forum 2008 (pp. 3-13), Cary, NC: SAS Institute Inc. Available at https://support.sas.com/resources/papers/sgf2008/gtl.pdf.

Matange, S. (2013). *Getting Started with the Graph Template Language in SAS: Examples, Tips, and Techniques for Creating Custom Graphs,* Cary,NC: SAS Institute Inc.

ACKNOWLEDGMENTS

I would like to thank my colleagues Jai Deep Mittapalli and Randall Nordfors for sharing their experience and helping with questions about GTL. I would like to sincerely thank my manager Balavenkata R Pitchuka and programming director Michiel Hagendoorn for reviewing this paper and providing valuable inputs.

RECOMMENDED READING

- Kuhfeld, Warren F. Mar 2016. *Basic ODS Graphics Examples*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/prod-p/grstat/9.4/en/PDF/odsbasicg.pdf
- Kuhfeld, Warren F. Nov 2015. *Advanced ODS Graphics Examples*. Cary, NC: SAS Institute Inc. Available at https://support.sas.com/documentation/prod-p/grstat/9.4/en/PDF/odsadvg.pdf
- *SAS® Product Documentation->SAS 9.2 Documentation, SAS 9.2 Output Delivery System User's Guide: Style Elements affecting Template-Based Graphics.* Available at http://support.sas.com/documentation/cdl/en/odsug/61723/HTML/default/viewer.htm#a003166123.htm
- SAS/STAT® User's Guide Customizing the Kaplan-Meier Survival Plot. Available at *https://support.sas.com/documentation/onlinedoc/stat/131/kaplan.pdf*
- *SAS® Product Documentation->SAS® 9.4 Graph Template Language: Reference, Fifth Edition*. Available at http://documentation.sas.com/?docsetId=grstatgraph&docsetTarget=grstatgraphwhatsnew94.htm&docsetVersion=9.4&locale=en
- Matange, Sanjay. 2016. *Clinical Graphs Using SAS®.* Cary, NC: SAS Institute Inc.
- Smith, Kevin D. 2013. *PROC TEMPLATE Made Easy: A Guide for SAS® Users*. Cary, NC: SAS Institute Inc.
- Harris, Kriss. 2017. "Hands-on Graph Template Language: Part A." Proceedings of *SAS® Global Forum. Orlando: SAS® Global Forum 2017*.

CONTACT INFORMATION

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

Jinit Mistry
Seattle Genetics, Inc.
(425) 527-2770
jmistry@seagen.com