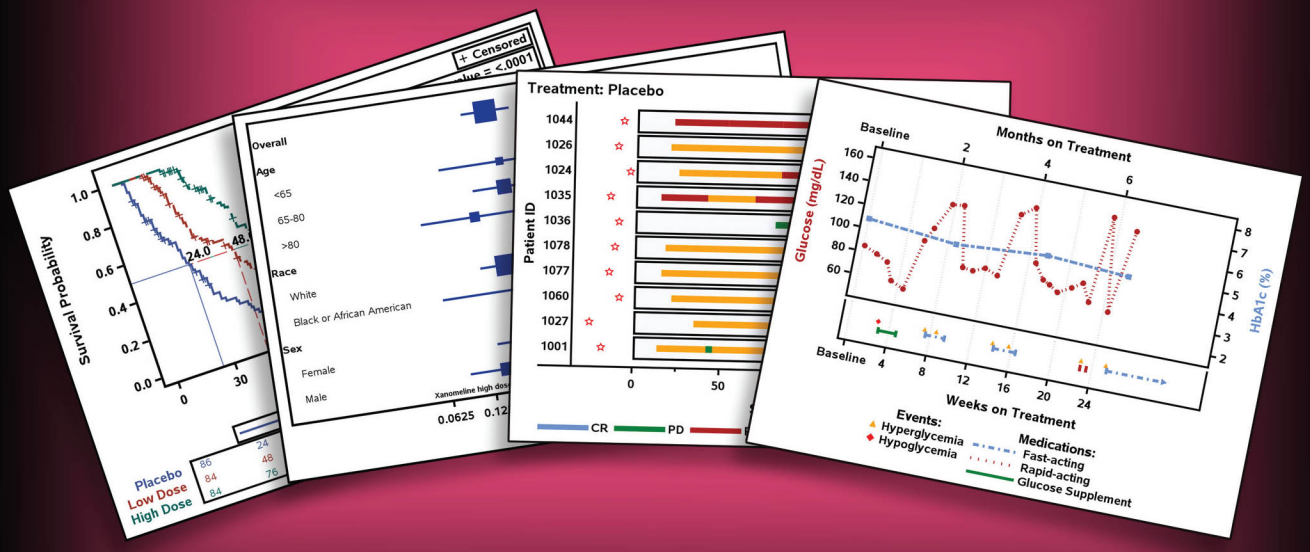
SAS® Graphics for Clinical Trials by Example



Kriss Harris
Richann Watson

The correct bibliographic citation for this manual is as follows: Harris, Kriss, and Richann Watson. 2020. *SAS® Graphics for Clinical Trials by Example*. Cary, NC: SAS Institute Inc.

SAS® Graphics for Clinical Trials by Example

ISBN 978-1-952365-99-7 (Hardcover)
ISBN 978-1-952365-95-9 (Paperback)
ISBN 978-1-952365-96-6 (Web PDF)
ISBN 978-1-952365-97-3 (EPUB)
ISBN 978-1-952365-98-0 (Kindle)

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

November 2020

Contents

About This Book

What Does This Book Cover?

As a programmer working in the health care and life science industries, you are sometimes required to create graphs of the clinical trial results. The graphs are typically used by statisticians in the review of the data, as well as for regulatory submissions. This book is intended to help illustrate how to create these various graphs. Although there are other books out there that show how to use the Statistical Graphics (SG) procedures and Graph Template Language (GTL), it is sometimes hard to translate that information into real-world use within the clinical industry. This book demonstrates through the use of examples, how to create some simple and straightforward graphs, as well as some complex graphs using CDISC ADaM data sets as the source in many examples. This book provides an introduction to the Output Delivery System (ODS) as well as a high-level overview of the basics of several SG procedures and GTL. This book does not go into all the SG procedures or all plots statements and options available. To cover all SG procedures, plots and available options are beyond the scope of this book.

Is This Book for You?

This book is designed to aid programmers working in the pharmaceutical industry to create graphs. Regardless if you are just learning how to produce graphs or have been working on graphs for a while. If you want to gain a better understanding of the reason why you are asked to produce graphs for clinical trials and want to know how to create specific types of graphs used for clinical trials, then this book is aimed at users like you.

What Are the Prerequisites for This Book?

In order to produce some of the graphs in this book, the data that you are working with might need to be manipulated so that it is in a format that can be used. Therefore, an understanding of Base SAS procedures and the DATA step is beneficial. Although not required, it can be useful to have a basic understanding of the SG procedures and GTL. Knowing how to create macro variables and macros would be advantageous.

What Should You Know about the Examples?

This book includes tutorials for you to follow and gain hands-on experience with SAS.

Software Used to Develop the Book's Content

All programs used to generate the graphs illustrated in this book are developed using SAS version 9.4 maintenance release 5.

Example Code and Data

The data used in this book is a combination of data from the CDISC pilot study, summary data sets based on the CDISC pilot study, and data created by the authors. The data from the CDISC pilot can be downloaded from the Updated Version of Pilot Submission Package (2013) https://www.cdisc.org/sdtmadam-pilot-project.

The data in Chapter 5 is fictitious data created by the authors. The authors are not oncology experts; however, they have worked on oncology studies where they had to produce time-to-event (TTE) graphs. Brief background information about TTE endpoints and types of graphs used are included to provide the reader with a high-level understanding of the type of data and output.

Also, in Chapter 6, the data is fictitious data created by the authors. The authors are not RECIST experts nor oncology experts. However, they have worked with both types of data and give a brief description and background on some concepts to help illustrate the type of outputs being created.

Chapter 8 utilizes a data set that contains two patients. Data for patient ABC-DEF-0001 is data based on one of the author's dogs who had an autoimmune disease, diabetes, and bladder cancer. The author had to vigilantly monitor blood work, events, and medications. Data for patient ABC-DEF-0002 is fictitious data, and it is not used in any of the examples.

The use of the word "patients" or "subjects" is typically determined based on the client. For this book, we opted to use the word "patients." However, there are some SAS procedures that produce default graphs that have the word "subjects" in a label.

Although all data sets are provided to help with executing the programs, they are not necessarily following standard CDISC ADaM data set naming conventions. Typically, these data sets would be temporary data sets created at time of graph creation. They are saved as permanent data sets to facilitate the execution of the programs that are found in the book.

CustomSapphire is a custom style template used specifically for the book, but was based on one of the standard SAS style templates - Sapphire. However, any of the standard SAS style templates could have been used.

All the data and code used in this book are available at https://github.com/rwatson724/SAS-Graphs-Clinical-Trials-Example.

You can access the example code and data for this book by linking to its author page at https://support.sas.com/authors.

Output and Graphics

All graphs shown in this book are created using the programs described in each chapter. However, in some instances, the full code is not available in the book in order to save space. The full code for each graph can be found https://github.com/rwatson724/SAS-Graphs-Clinical-Trials-Example.

We Want to Hear from You

SAS Press books are written *by* SAS Users *for* SAS Users. We welcome your participation in their development and your feedback on SAS Press books that you are using. Please visit sas.com/books to do the following:

- Sign up to review a book
- Recommend a topic
- Request information on how to become a SAS Press author
- Provide feedback on a book

Do you have questions about a SAS Press book that you are reading? Contact the author through saspress@sas.com or https://support.sas.com/author_feedback.

SAS has many resources to help you find answers and expand your knowledge. If you need additional help, see our list of resources: sas.com/books.

About These Authors

Kriss Harris worked at GlaxoSmithKline as a statistician supporting drug discovery. At GSK, he developed an increasing passion for both SAS graphics and teaching and taught SAS graphics to SAS programmers, statisticians, and scientists. After leaving GSK, he became an independent statistical programmer and has consulted at Eli Lilly, Eisai, and MedAvante-ProPhase. Currently, Kriss is consulting at Eli Lilly supporting drug reimbursements within the oncology therapeutic area and at MedAvante-ProPhase supporting the construction of edit checks. Kriss is based in London, England, and holds a bachelor's degree in statistics and internet computing from Kingston University and a master's degree in statistics from the University of Sheffield.
https://www.linkedin.com/in/krissharris/
https://krissharris.co.uk/

Richann Watson is an independent statistical programmer and CDISC consultant based in Ohio. She has been using SAS since 1996 with most of her experience being in the life sciences industry. She specializes in analyzing clinical trial data and implementing CDISC standards. Additionally, she is a member of the CDISC ADaM team and various sub-teams.
Richann loves to code and is an active participant and leader in the SAS user group community. She has presented numerous papers, posters, and training seminars at SAS Global Forum, PharmaSUG, and various regional and local SAS user group meetings. Richann holds a bachelor's degree in mathematics and computer science from Northern Kentucky University and master's degree in statistics from Miami University.
https://www.linkedin.com/in/richann-watson-31435422/
https://www.linkedin.com/company/28145660/admin/
https://datarichconsulting.com/
Learn more about these authors by visiting their author pages, where you can download free book excerpts, access example code and data, read the latest reviews, get updates, and more:
http://support.sas.com/harris
http://support.sas.com/watson

Chapter 1: Introduction to Clinical Graphics

1.1 Introduction to Output Delivery System Statistical Graphics

There are several procedures in SAS that use the Output Delivery System (ODS) Statistical Graphics software. Part of the ODS Graphics are Statistical Graphics (SG) procedures and Graph Template Language (GTL). SG procedures generate plots that are based on procedure-driven GTL templates. In other words, the templates are determined based on the plot statements and options provided within the SG procedure. In addition, you can generate graphs from several analytical procedures, such as the LIFETEST procedure. With ODS Graphics, you are able to obtain graphs automatically through the analytical procedures or create custom graphs using SG procedures or GTL (Matange, 2016).

Knowing ODS Graphics enables you to modify or add features that are based on the procedure-driven templates as well as create customized graphs that might not otherwise be possible. GTL makes it easier to incorporate features, such as embedding a table of data within the output area or displaying multiple graphs, regardless of whether the graphs are of the same or different type on the same page. Prior to GTL, these features might have been difficult to achieve. Furthermore, according to Matange (2013) some additional reasons to learn GTL are:

- GTL provides in one system the full set of features that you need to create graphs from the simplest scatter plots to complex diagnostics panels.
- GTL is the language used to create the templates shipped by SAS for the creation of the automatic graphs from the analytical procedures. To customize one of these graphs, you will need to understand GTL.
- GTL represents the future for analytical graphics in SAS. New features are being added to GTL with every SAS release.

The following conventions are used in the text (and not the SAS code) within this book. Terms that are in all capital letters denote SAS procedures, SAS statements, data sets, and variables in a data set (for example, PROC SGPLOT, the SCATTER statement, the ADAE data set, and the

AESTDT variable). Terms that are in italic capital letters denote macro variables (for example, *SYSDATE* and *SYSTIME).* Bold italic letters denote macros (for example, **%SurvivalTemplateRestore**).

1.1.1 Understanding the Basic: Statistical Graphics Procedures

SG procedures can be used to create a number of graphs. They use clear and concise syntax. SG procedures include these two procedures: SGPLOT and SGPANEL. Although there are other SG procedures, SGPLOT and SGPANEL are the only SG procedures that are used in the production of some of the graphs illustrated in the book. The SGRENDER procedure is used along with the TEMPLATE procedure to produce custom graphics. These two procedures combined are referred to as GTL in this book.

1.1.1.1 Overview of the SGPLOT Procedure

For many graphs, the SGPLOT procedure is typically sufficient. In order to use SGPLOT at least one plot statement must be specified. Within SGPLOT you can overlay different plots so that they occupy the same data area in the graph. Types of graphs that can be produced with SGPLOT include bar charts, high-low plots, block plots, series plots, and waterfall plots.

Although only one plot statement is needed, further control over the graph's appearance can be achieved with the use of additional statements. You can control various aspects such as colors and markers that are associated with a style with the use of STYLEATTRS. If there is a particular Unicode character or an image that you want to use in one or more plots specified within the SGPLOT procedure, you can define the marker symbol using the SYMBOLCHAR or SYMBOLIMAGE. In addition, you can control the axes options with the use of XAXIS, X2AXIS, YAXIS, and Y2AXIS. XAXIS and X2AXIS control the X (bottom X axis) and X2 (top X axis) respectively; while YAXIS and Y2AXIS control the Y (left Y axis) and Y2 (right Y axis) respectively.

Each statement within SGPLOT has additional options that allow for further control of the appearance. The options vary based on the plot specified, but you can change other attributes like color, text rotation, fill pattern, and line pattern. The syntax for SGPLOT is simple as illustrated in Program 1-1.

Program 1-1: SGPLOT Procedure Syntax

```
proc sgplot <options>;
   <… plot statements …> / <options>;
run;
```

1.1.1.2 Overview of the SGPANEL Procedure

With the SGPANEL procedure, you can create a panel of similar graphs based on one or more classification variables. In order to use SGPANEL, the PANELBY statement must be specified. The PANELBY statement determines the different combinations of the classification variables. At least one classification variable needs to be provided when using SGPANEL. With SGPANEL, SAS automatically determines the panel layout, that is, SAS determines the number of individual cells

for a data area in the graph, and, if necessary, SAS creates multiple graphs if the number of unique combinations of classification variables does not fit on one graph.

Other than the use of PANELBY, SGPANEL can use the same plot statements and options to help control the appearance of each individual cell in the graph. Program 1-2 demonstrates the standard syntax for SGPANEL.

Program 1-2: SGPANEL Procedure Syntax

```
proc sgpanel <options>;
   panelby variable / <options>;
   <… plot statements …> / <options>;
run;
```

1.1.1.3 Overview of the SGSCATTER Procedure

While the SGSCATTER procedure can be used to create a simple scatter plot of Y by X, it can also be used to create a panel of scatter plots that are based on the different combinations of variables specified. The type of panel created is determined based on whether the PLOT, COMPARE, or MATRIX statement is used. The use of PLOT, COMPARE, or MATRIX statement is required when using SGSCATTER. In addition, each one of the statements have their own set of options that enable you to control the appearance of the plots being produced.

To produce a simple scatter plot similar to what would be produced if you used SGPLOT with the SCATTER plot statement, you would use the PLOT statement and specify your Y and X variables. The PLOT statement creates a cell for each combination of Y variables and X variables. For example, if there are two Y variables and one X variable specified, then two cells are created – one for the first Y variable by the X variable and another for the second Y variable by the X variable.

With the use of the COMPARE statement, you can achieve similar results as with the PLOT statement, with the main difference being that the COMPARE statement produces a panel with shared axes, while the PLOT statement has individual axes for each X and Y variable combination.

The MATRIX statement produces a matrix of scatter plots. The number of cells within the matrix is determined by the number of numeric variables that are specified. The downward diagonal from left to right contains the variable name or label.

When specifying more than one X or Y variable, the variables should be enclosed in parenthesis and should be separated by a space. Refer to Program 1-3 for general syntax for SGSCATTER for PLOT, COMPARE, and MATRIX.

Program 1-3: SGSCATTER Procedure Syntax

```
proc sgscatter <options>;
   plot ( y-variable(s) ) * ( x-variable(s) );
run;

proc sgscatter <options>;
   compare x = ( x-variable(s) )
           y = ( y-variable(s) ) / <options>;
run;

proc sgscatter <options>;
   matrix numeric-variable(s) /<options>;
run;
```

1.1.2 Understanding the Basics: Graph Template Language

While SG procedures do produce stellar graphs, there are times when you might want to customize specific aspects of the graph or you might need to produce a graph that is not easily done using the SG procedures.

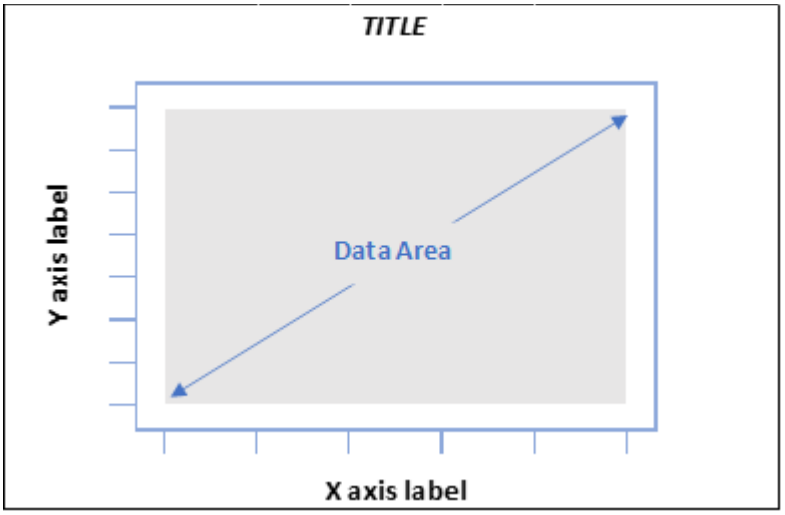
In order to use GTL, it is important that you first understand the basics. This includes understanding the difference between the graphics output area and the procedures output area. In addition, you should have an understanding of the difference between single-cell and multi-cell graphs and the various layouts associated with each. You should also have knowledge of the basic syntax for the most common layouts.

1.1.2.1 Draw Spaces for a Graph

Before you start using GTL, you need to know the difference between the types of drawing space in a graph. There are four different drawing spaces: data, wall, layout, and graph.

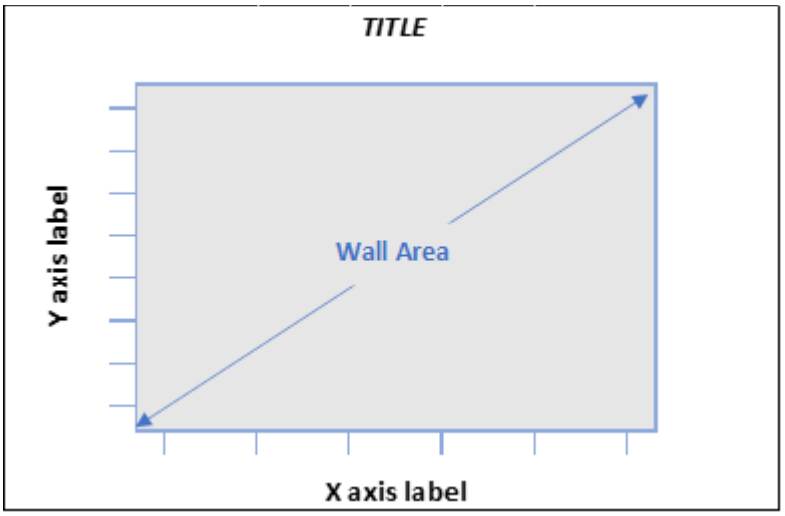
The data area is the innermost portion of the output as shown in Figure 1-1. This is the portion of the graph that contains the data, that is, the portion where the plots are displayed. While most of the graphs produced have a data area, there are a few that do not. “The data area does not apply to graphs that do not have axes, such as pie charts” (SAS Institute Inc., 2016). Graphs without axes, such as pie charts, are drawn using a REGION layout.

Figure 1-1: Data Area of Graph



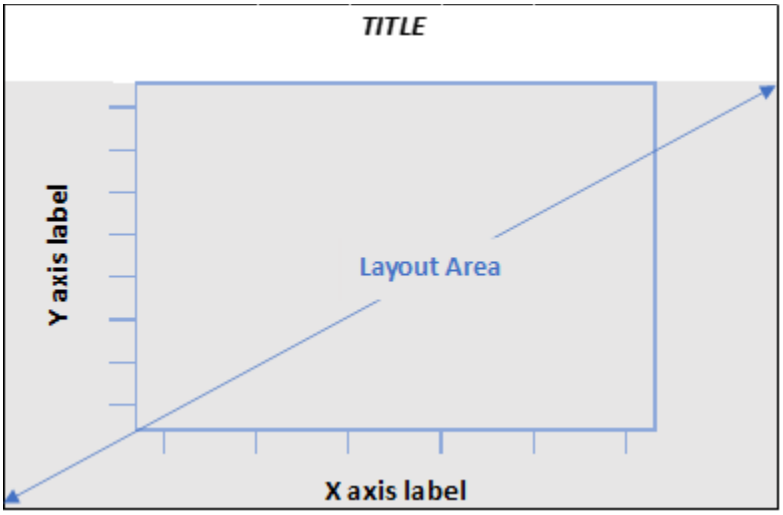
Notice how the data area does not include the space up to the axes. This space is part of the wall area, which also encompasses the data area, as illustrated in Figure 1-2. The wall area is bound by the X and Y axes and X2 and Y2 axes if they are used. Similar to the data area, if there are no axes, then there is no wall area.

Figure 1-2: Wall Area of Graph



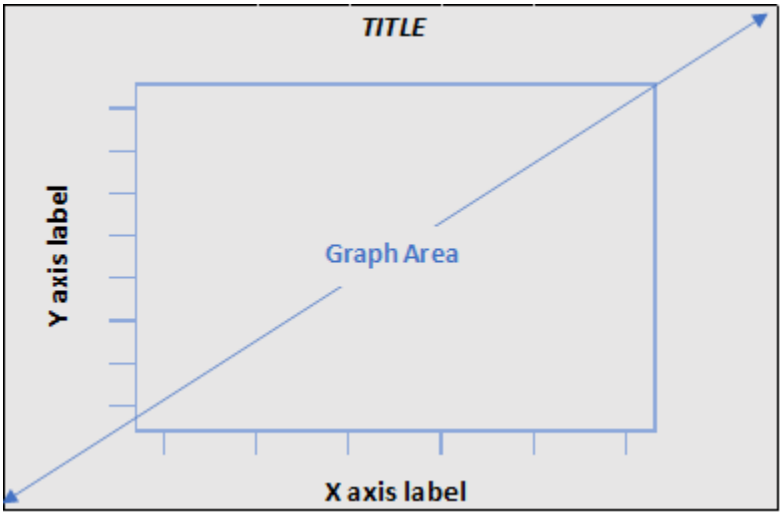
The part of the graph that includes additional information such as axis-labels, tick marks and legends is in the layout area. Any information that is displayed on the graph that is defined within a LAYOUT block is part of the layout area as seen in Figure 1-3.

Figure 1-3: Layout Area of Graph



The final area is the graph area, and this encompasses the entire graph as shown in Figure 1-4.

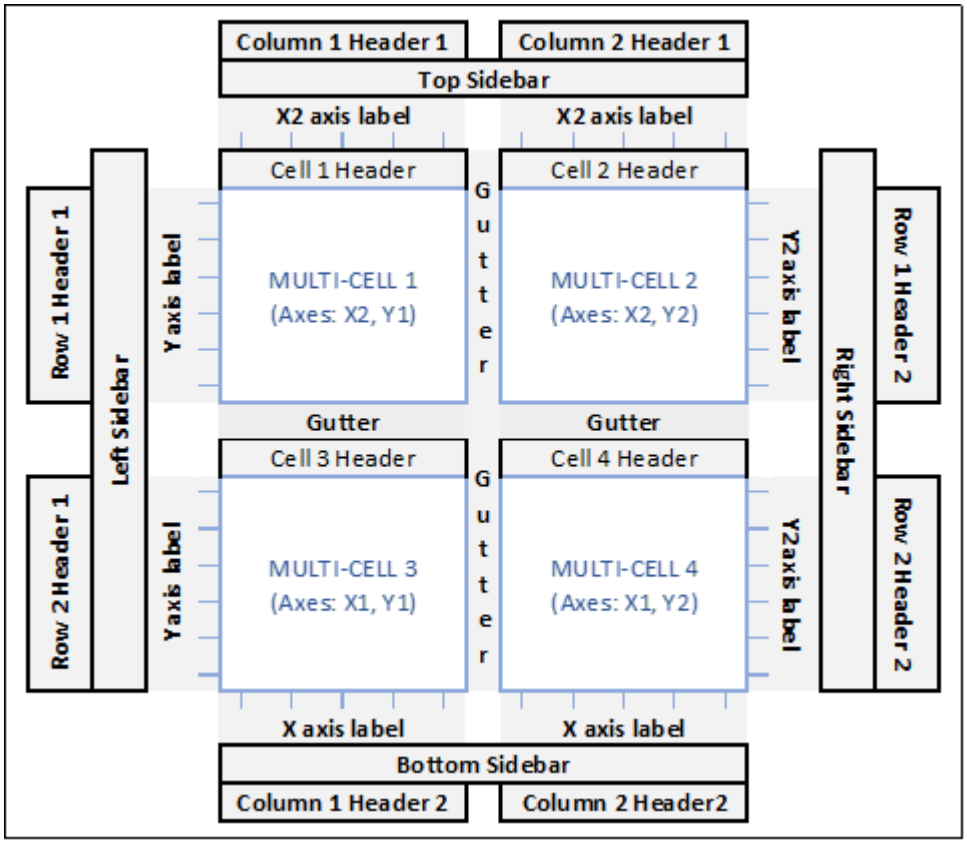
Figure 1-4: Graph Area of Graph



Figures 1-1 through 1-4 illustrate the various areas for a single-cell graph. When the plots occupy the entire data area, whether there is a single plot or multiple plots that are overlaid (for example, multiple GTL plot statements within an overlay so that the plots appear on top of each other within the same data area), this is referred to as a single-cell graph. When each plot occupies a different data area, this is known as a multi-cell graph. Multi-cell graphs can either be pre-defined or data-driven. A pre-defined multi-cell graph breaks the graph area into pre-defined portions so that each portion represents different pieces of information that include the data, axes, headers, and titles. The data-driven multi-cell graph breaks the graph area into as many parts necessary based on the data.

As the graph becomes more complex when there are multiple cells within the same page, the graph area might become more complex with the incorporation of gutters, cell headers, and sidebars, as seen in Figure 1-5.

Figure 1-5: Complex Graph Area



1.1.2.2 Layouts for GTL

There are several types of layouts available in GTL. What needs to be graphed determines the type of layout. These layouts can be classified into one of three categories mentioned previously, which are: single-cell, multi-cell pre-defined, and multi-cell data-driven.

Within the single-cell graph, there are five different layouts. All single-cell layouts, with the exception of OVERLAY3D, are 2-D plots.

- **OVERLAY:** General layout that allows for the plotting of one or more 2-D plots in the same cell provided all plots can share the same axes type.
- **OVERLAYEQUATED:** Similar to OVERLAY with the differences being that the axes can only be linear, the display distance on both axes is the same for equal data, and the aspect ratio of the plot display and plot data are equal.

- **PROTOTYPE:** Layout that creates a prototype of a plot that will be used repeatedly based on the data. Since the number of repeats is based on the data, it can only be used with DATAPANEL or DATALATTICE.
- **REGION:** General plot that does not use axes.
- **OVERLAY3D:** General layout that allows for the plotting of one or more 3-D plots in the same cell provided all plots can share the same axes type.

For pre-defined multi-cell graphs, there are two layouts. Both layouts are 2-D plots.

- **GRIDDED:** A simple multi-cell layout with all cells pre-defined. The grid is determined based on the number of rows and columns that are specified in the options. The cells have the same proportion in regard to height and width and how much space is allocated for the cell content is automatically determined.
- **LATTICE:** A complex multi-cell layout with all cells pre-defined. Similar to GRIDDED, the grid is determined by the number of rows and columns specified in the options. LATTICE is very flexible in that it allows each cell to have different heights and widths. The size of the cells can be specified by indicating the width of each column (COLUMNWEIGHTS) or the height of each row (ROWWEIGHTS) or indicating both width and height. The size of the cells can also be automatically determined based on whether or not the width or height should be equally divided among all columns or among all rows, respectively. In addition, the width or height can be the maximum preferred width from a vertically one-dimensional plot or maximum preferred height from a horizontally one-dimensional plot with the other columns and rows that do not contain the vertically or horizontally one-dimensional plot having an equal width or height from the remaining space.

Both GRIDDED and LATTICE layouts can be used with OVERLAY layout in order to create other types of layouts by nesting within each cell.

Multi-cell data-driven graphs also have two layouts that are both 2-D plots.

- **DATAPANEL:** Uses the PROTOTYPE layout to display a panel of similar graphs that are based on the data. The number of cells produced is determined by the crossings of n classification variables. In order words, each unique value for each classification variable is paired with each unique value of the other classification variable. For example, if you have two classification variables and one variable has three levels and the other variable has four levels, then there would be 12 crossings, that is, each level in the first variable is crossed with each level in the second variable.
- **DATALATTICE:** Similar to DATAPANEL with the difference being that the number of cells is based on crossings of 1 or 2 classification variables.

In addition, there are other types of layouts that do not necessarily pertain to the number of cells.

- **GLOBALLEGEND:** This allows for the creation of one legend using multiple discrete legends or merged legends placed at the bottom of the graph. A continuous legend is not supported within a GLOBALLEGEND. In addition, there is only one GLOBALLEGEND statement allowed per template.
- **INNERMARGIN:** Can only be used inside an OVERLAY or PROTOTYPE layout. It reserves space within those two layouts to provide a nested block plot or axis table. Note that only a BLOCKPLOT or AXISTABLE statement can be used within an INNERMARGIN.

With the exception of OVERLAYEQUATED, OVERLAY3D and REGION, there is at least one example illustrating the different layouts throughout the book.

1.1.2.3 General Syntax for GTL

Before discussing the general syntax for the layouts, you need to understand the syntax for the TEMPLATE procedure. All templates created for the purpose of producing a graph will have the structure illustrated in Program 1-4.

Matange (2013) points out that using GTL to create a graph is a two-step process:

1. First, you need to define the structure of the graph using the STATGRAPH template. In the creation of the template, no graph is actually produced.
2. Second, you need to associate the data in order to render the template that will produce the graph.

Program 1-4 illustrates the basic syntax of the TEMPLATE procedure, the first step in using GTL, with a brief explanation of each component provided.

Program 1-4: TEMPLATE Procedure Syntax

```
proc template;
 define statgraph templatename;    ❶
   begingraph / <options>;    ❷
         layout type / <options>;    ❸

            <… GTL SAS code …>    ❹

         endlayout;    ❸
   endgraph;    ❷
 end;    ❶
run;
```

❶ The first step to creating a custom graph is by defining a STATGRAPH template within PROC TEMPLATE. This allows for the "opening" of the template that you use to define your graph. Each custom template starts with "define statgraph ***templatename***" that enables you to provide a name for the custom template. The template name is used when rendering the graph. Since you opened the template, you must also close the template, in order to do so you need a corresponding END.

❷ Each STATGRAPH definition has at most one BEGINGRAPH statement, which is the signal that indicates the various components of the custom template are specified within the block. BEGINGRAPH has a corresponding ENDGRAPH statement, which signals the end of the graph template definition.

❸ LAYOUT enables you to specify the type of layout you want and assign the necessary options. Layouts can be nested depending on whether LATTICE, GRIDDED, DATAPANEL or DATALATTICE is specified in the statement. For each LAYOUT, you need to signal the end of the layout with ENDLAYOUT. General syntax on the layouts is explained in Section 1.1.2.4 *General Syntax for Layouts*.

❹ The majority of the work takes place within the LAYOUT block(s). The type of layout(s) and plot(s) drives how this portion is programmed. If there are multiple layouts or nested layouts, then there needs to be a corresponding ENDLAYOUT statement for each. As many plot statements and their corresponding options needed to build the graph should be encompassed within the corresponding LAYOUT.

The template created describes the graph. It is a program that is stored and compiled to be accessed later. As mentioned in "What's Your Favorite Color? Controlling the Appearance of a Graph" (Watson, 2020), the templates that are created by default are stored in SASUSER.TEMPLAT. However, you can make the template available to others by saving it to a permanent location by modifying the ODS PATH statement.

Program 1-5: Saving a Template to a Permanent Location

```
libname ADAM "C:\Users\gonza\Desktop\GTL_Book_Harris_Watson\Data\ADaM";

ods path ADAM.TEMPLAT (update) SASUSER.TEMPLAT (read) SASHELP.TMPLMST (read); ❶
```

❶ In order to save the graph template that you created to a permanent location for others to use, the ODS PATH statement needs to be updated so that it lists the location of where it should reside. SAS searches the template stores in the order in which they are listed. If no modification is made to the ODS PATH, then SAS will search in SASUSER.TEMPLAT first and then SASHELP.TMPLMST. The UPDATE option allows new templates to be written to that template store as well as allowing templates to be read from it.

As stated earlier, creating a graph with GTL is a two-step process because the template itself does not actually produce a graph. Thus, the second step is to render the graph. This is done by associating the data with the template in the SGRENDER procedure. It does not matter what the data source is as long as the data has all the necessary variables that are referenced in the template.

Program 1-6: Associating the Data with the Graph Template

```
proc sgrender data = datasetname ❶  template = templatename;  ❷
   <optional SAS statements>;
run;
```

❶ The name of the data set that contains all the necessary variables that are to be associated with the graph template.

❷ The name of the graph template that will be rendered to produce the actual graph. If the template is saved to a permanent location, then the ODS PATH should have the location specified.

Below are a few examples of other SAS statements that can be used in the SGRENDER procedure.

- BY statement to allow rendering by different groups.
- FORMAT statement to allow data to be formatted without losing the order of the data. This can be beneficial if the data needs to be in a specific order, but if you were to sort by the labels, you would lose the ordering.
- LABEL statement to allow X and Y-axis labels to be defined if they are not defined in the template. This could be beneficial if the template is used for producing multiple outputs but with different X and Y-axis labels.

1.1.2.4 General Syntax for Layouts

PROC TEMPLATE enables you to customize your SAS output. The customization can include things such as color and fonts, which is part of the style. In addition, PROC TEMPLATE is used to specify the necessary GTL statements in order to define the structure of the graph. This includes specifying the layout(s). Each layout is initiated with a LAYOUT statement and terminated with an ENDLAYOUT statement regardless of which layout you use.

The syntax for all the layouts except for the two that are data-driven is similar but will have varying options. The general syntax is simple, as illustrated in Program 1-7.

Program 1-7: General Syntax for Majority of the Layouts

```
layout type </options>;
   <… GTL SAS code …>
endlayout;
```

The two layouts that are data-driven require that the PROTOTYPE layout be nested within the data-driven layouts as illustrated in Program 1-8 and Program 1-9.

Program 1-8: General Syntax for DATALATTICE

```
layout DATALATTICE rowvar ❶ = class-variable
                   columnvar ❶ = class-variable </option(s)>;
   layout prototype </options>;  ❷
      <… GTL SAS code …>
   endlayout;
endlayout;
```

❶ Either ROWVAR and/or COLUMNVAR must be specified when using DATALATTICE. Both can also be specified. ROWVAR represents the classification variable for the rows, while COLUMNVAR represents the classification variable for the columns.

❷ The PROTOTYPE layout is used to create the structure of the graph that is used repeatedly based on the data. The general syntax for PROTOTYPE follows the same syntax as shown in Program 1-7.

Program 1-9: General Syntax for DATAPANEL

```
layout DATAPANEL classvars = (class-var1...class-varn)  ❶ </option(s)>;
   layout prototype </option(s)>;
      <… GTL SAS code …>
   endlayout;
endlayout;
```

❶ CLASSVARS is a list of the classification variables that are to be used to generate the panel. A cell is created for each crossing of the classification variables.

1.1.2.5 Syntax for LATTICE Layout

While the general syntax for LATTICE is similar to the others in its simplest state, as shown in Program 1-7, depending on the desired graph, the syntax used can be more in-depth. LATTICE allows for different axes for each portion of the graph but also allows for the sharing of axes. When using the LATTICE layout, by default, each cell within the lattice has its own internal axes. However, if your cells within the lattice have a common scale, then you can create row and/or column axes that the various cells can share. Program 1-10 illustrates the syntax that would be used when you want to control the axes.

Program 1-10: Controlling Axes Syntax for LATTICE

```
layout LATTICE </options>;
   <… GTL SAS code …>
   <columnaxes </options>;   ❶
       columnaxis / axis-option(s);
       <… GTL SAS code …>
   endcolumnaxes;>
   <column2axes </options>;   ❶
      columnaxis / axis-option(s);
       <… GTL SAS code …>
   endcolumn2axes;>

   <rowaxes </options>;   ❷
      rowaxis / axis-option(s);
       <… GTL SAS code …>
   endrowaxes;>
   <row2axes </options>;   ❷
      rowaxis / axis-option(s);
       <… GTL SAS code …>
   endrow2axes;>

   <columnheaders;   ❸
       <… GTL SAS code …>
   endcolumnheaders;>
   <column2headers;   ❸
       <… GTL SAS code …>
   endcolumn2headers;>
```

```
   <rowheaders;    ❹
      <… GTL SAS code …>
   endrowheaders;>
   <row2headers;    ❹
      <… GTL SAS code …>
   endrow2headers;>

   <sidebar </options>;    ❺
      <… GTL SAS code …>
   endsidebar;>
endlayout;
```

❶ COLUMNAXES and COLUMN2AXES are used to control the X and X2 axes, respectively. However, if all parts of the graph are to have the same axes, then only one set of axes statements would need to be defined.

❷ ROWAXES and ROW2AXES are used to control the Y and Y2 axes, respectively. Similar to COLUMN(2)AXES, if all parts have the same axes, then only one set needs to be defined.

❸ COLUMNHEADERS displays the header that is to be displayed at the bottom of the graph. This is considered the primary header. In addition to COLUMNHEADERS, you can have COLUMN2HEADERS, which is the secondary header that is displayed at the top.

❹ Like COLUMNHEADERS, ROWHEADERS displays the primary header on the left side of the graph while ROW2HEADERS, which displays the secondary header on the right side.

❺ SIDEBAR is useful for adding information that can span across columns or rows, and up to four sidebars can be specified, one for the bottom, top and each side. For example, you can use the SIDEBAR statement to specify a text that spans across the columns or rows, or it can be used to display legends.

1.2 Reason Why Clinical Graphs Are Needed and What They Should Convey

The majority of the output produced for clinical data is provided in the form of a listing or table. Tables can contain things such as summary statistics over time or confidence intervals as well as *p*-values to test if the difference between two groups of data are significant. While this information is nice in a tabular form, sometimes it does not fully convey the message that you want to convey. A lot of individuals are more apt to understand the meaning of something if it is displayed as a picture. In other words, a lot of people are more visual. They want to see pictures to get a better grasp of what the message is. Because of this, sometimes it is better to display clinical data in a graph.

With a clinical graph, you can better understand mean values over time. In addition, with the clinical graph, you can overlay different types of summary statistics to get a better "picture." For example, you can have a series plot of the mean change from baseline over time with whiskers that show +/- X standard deviations from the mean. This not only shows you how the mean value is changing over time but also shows the spread of each mean value. Granted, all this information can be captured in a table format, but reading each number and flipping through numerous pages makes it easier to miss a key point. However, with the visual representation,

the eye is immediately drawn to the part that does not seem to be in alignment with the rest of the data. In addition, with a graphical representation of the data, you can showcase results of different data on one page. For example, as illustrated in Chapter 8, you can overlay medication usage with the change of a particular lab value over time to help monitor if the medication had an effect on the lab result.

Regarding what you should convey in a clinical graph, it depends on the message you want to deliver. Thus, when determining what type of graph you want, you need to ask yourself, "What is the story I am trying to tell?" If you are looking to see the effect that a medication has on particular lab values, then you would need to ensure you have a graph that shows the lab values over time with the medication taken plotted on the same graph. Or, if you are interested in specific adverse events (AEs) and the lab results that are related to those adverse events, then you need to think about the best way to illustrate that. For example, would you display that as a bar chart of the number of AEs at different levels of the mean lab result or would you display that as a box plot of the mean lab values with a scatterplot of AEs overlaid?

1.3 Chapter Summary

While SAS has procedures that produce nice graphs, understanding the basics of ODS Graphics can help you to modify those features that you need to change in order to create the desired output or to help you produce truly custom graphs. You have the ability to modify the template from the procedure-defined templates or create your own graph template. If you know what type of message you want to convey, you can create clinical graphs that will help with understanding your data.

References

Harris, K., & Watson, R. (2018). "Great Time to Learn GTL." Proceedings of the Annual Pharmaceutical SAS Users Group Conference. Seattle: PharmaSUG.

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.

Matange, S. (2016). *Clinical Graphs Using SAS®.* Cary, NC: SAS Institute Inc.

SAS Institute Inc. (2016). *SAS® 9.4 Graph Template Language: User's Guide, Fifth Edition.* Cary, NC: SAS Institute Inc. Retrieved from https://documentation.sas.com/api/docsets/grstatug/9.4/content/grstatug.pdf?locale=en

Watson, R. (2019). "Great Time to Learn GTL: A Step-by-Step Approach to Creating the Impossible." Proceedings of the *SAS Global Forum 2019 Conference* (p. 1). Cary, NC: SAS Institute Inc. Dallas: SAS Institute Inc.

Watson, R. (2020). "What's Your Favorite Color? Controlling the Appearance of a Graph." Proceedings of the *SAS Global Forum 2020 Conference.* Cary, NC: SAS Institute Inc.