PharmaSUG 2022 - Paper DV-74

Metadata Driven Approach for Creation of Clinical Trial Figures

Jeremy Gratt, Modular Informatics LLC, Thornwood NY;
Qiuhong Jia, Seagen Inc., Bothell WA;
Girish Kankipati, Seagen Inc., Bothell WA.

ABSTRACT

In clinical studies, a variety of figures is required to visually analyze the responses to study drug. Creation of these figures presents many programming challenges. For example, SAS® provides multiple figure creation methods and this makes it difficult for programmers to learn. Modern figures used for clinical trials often overlay multiple kinds of data within a single graph, and this complicates the input data structure used to generate the figures. Within SAS®, different kinds of figure elements (lines, bars, markers) have different rules for use, and these differences can also be tricky to learn. Creation of legends that combine and describe the various figure elements can present challenges that are not always natively handled within SAS®. Finally, clinical study teams may ask for study-specific customizations to describe specific trial designs.

In the paper we describe these various challenges to figure generation and present our solution for creation of standard macros for spider plots, waterfall plots and swimmer plots that can be used across clinical programs. This solution includes a novel use of metadata provided by programmers at the study level that allows for customization of any number of figure elements (bars, lines, markers), flexibility towards handling of input data of varying structures, and custom creation of complex legends. We also discuss how the use of metadata allows for standardization of the figure elements across studies and clinical programs.

INTRODUCTION

For construction of standard macros to generate clinical trial figures, a standard approach to the setup of the figure macros was devised:

- A separate macro is developed for each type of figure (e.g., separate macros for waterfall, spider plot and swim plot macros).
- The figure macros focus on the actual drawing of the figure; they will not perform any numerical calculations and instead use the data defined in the input data set. Thus, it is up to the programmer to perform any required calculations or merging/joining in order to prepare the input data set that the macro uses; however, the macro accepts standard ADaM structures and the data manipulation is expected to be relatively minor.
- The figure macros are comprised of a single macro call to generate the figure. Other designs may include multiple helper macros that programmers combine together to generate a figure, but for our purposes we chose to design each macro call to generate one (and only one) figure. Generation of multiple figures is handled by multiple calls to the macro.
- There are multiple ways to generate figures in SAS, and we standardized on using PROC TEMPLATE in combination with PROC SGRENDER.
- All figure attributes relating to line, bar, and marker symbol styles are be defined using metadata. This allows users to define any number of figure elements using distinguishing colors/patterns/thickness/symbol types. The metadata approach makes it possible to define and reuse consistent figure attributes across multiple studies.

The following sections will describe how this approach was implemented with a focus on how metadata is used to define the figure elements for waterfall, spider, and swimmer plots. We will also describe the SAS macro programming techniques used to harness the metadata to generate the figures.

FIGURE MACRO INPUTS AND OUTPUTS

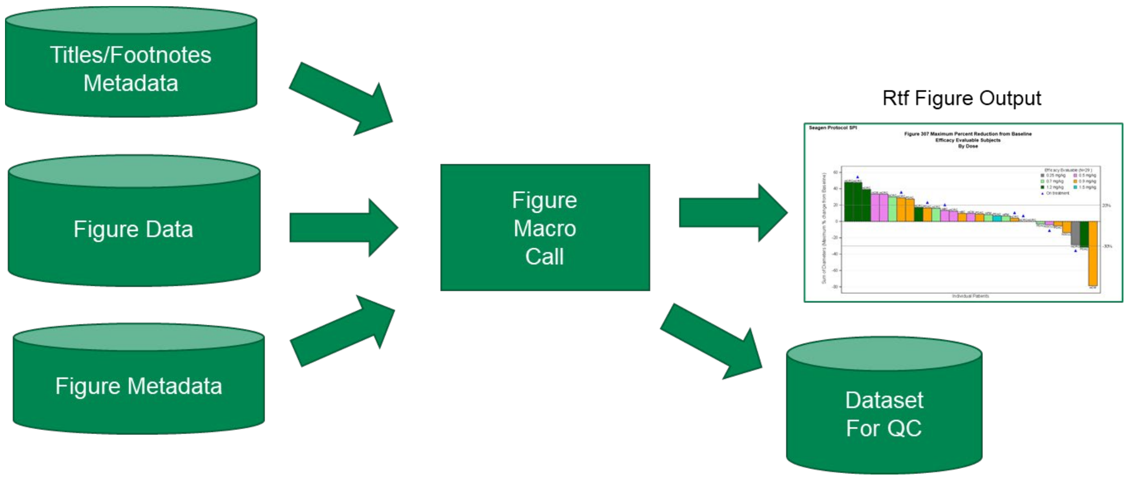


Figure 1. Figure Macro Inputs and Outputs

Notes about the figure macro inputs and outputs:

- Input Titles/Footnotes Metadata: the macros use titles and footnotes defined in a departmentally standardized spreadsheet used for storing this type of information.
- Input Figure Data: It is up to the programmers to prepare this input data that contains variables for the plotted lines, bars, and markers.
- Input Figure Metadata: as will be shown below, the figure metadata defines figure attributes (colors, symbols, etc.) for lines, bars, and marker elements within the figure.
- Output RTF: the Figure macro generates an RTF output file containing the figure. This may be portrait or landscape, of any size.
- Output Dataset for QC: The macro outputs a copy of the input data as a QC data set following a standardized naming convention. This can be used as part of the QC programming process to check the results.

MAIN MACRO CALL

The main call of the macro includes parameters for:

- IN_DS: name of the input data set. This is the data the programmer must prepare that contains the variables to be plotted. This data set is passed to the SGRENDER procedure in the macro.
- RESULT_VAR: macro parameter with the name of the result variable to be plotted. For example, if percent change variable PCHG is to be plotted then RESULT_VAR=PCHG. Applies to figure macros that plot a result variable like spider or waterfall plots.
- IN_BAR_ID or IN_META_BAR_DS: these macro parameters point to the metadata for bars in the figure (bar color, pattern, label, where statement to select the data). Bar metadata may be stored either in the standard study lookup folder using IN_BAR_ID or may be stored in a simple data set created in the calling program (using IN_META_BAR_DS).
- IN_MARKER_ID or IN_META_MARKER_DS: these macro parameters point to the metadata for markers in the figure (e.g., symbol, color, label, where statement to select the data).

- IN_LINE_ID / IN_META_LINE_DS: these macro parameters point to the metadata for lines in the figure (e.g., color, label, where statement to select the data).
- LEGEND macro parameters: to define legend title, font size, label.
- AXIS parameters: including axis definition, label, font size.
- DESIGN_WIDTH, DESIGN_HEIGHT: dimensions for the overall figure.

Here is an example of the waterfall macro call. Spider plot and swim plot macro calls are similar.

```
%mcr_spi_fig_waterfall(
    base_output_name = f-xxx,
    in_ds            = adtr,
    result_var       = pchg,
    in_bar_id        = wfbar1,
    in_marker_id     = wfmark1
    yaxis_label      = Sum of Diameters (Maximum % Change from Baseline),
    xaxis_label      = Individual Patients,
    yaxis_values     = -80 to 100 by 20,
    legend_title     = Efficacy Evaluable (N=&n.),
    design_width     = 8.5,
    design_height    = 4.2,
    graphborder_yn   = N
);
```

METADATA FOR THE WATERFALL MACRO

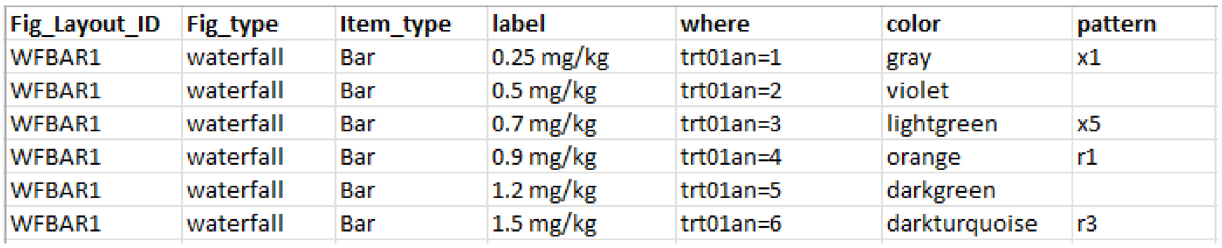
The following section contains details regarding the waterfall plot macro.

A waterfall plot displays the maximum reduction or minimum increase from baseline in sum of diameters. Each vertical bar represents one subject in the study. The markers on top of the bars define subject characteristics or milestones.

In waterfall plots, each bar represents one subject and corresponds to one row in the in_ds data set. The corresponding bar metadata contains one row per bar grouping (e.g., treatment). Users may define any number of bar types with different colors or patterns and may define any type of bar markers with different symbols or characters. The direction and height of bars depend on the result variable in the in_ds data set.

METADATA FOR WATERFALL BARS

Metadata for bars has the following structure. Each row of the metadata is used to define bar attributes to the rows of the input data:



Fig_Layout_ID	Fig_type	Item_type	label	where	color	pattern
WFBAR1	waterfall	Bar	0.25 mg/kg	trt01an=1	gray	x1
WFBAR1	waterfall	Bar	0.5 mg/kg	trt01an=2	violet	
WFBAR1	waterfall	Bar	0.7 mg/kg	trt01an=3	lightgreen	x5
WFBAR1	waterfall	Bar	0.9 mg/kg	trt01an=4	orange	r1
WFBAR1	waterfall	Bar	1.2 mg/kg	trt01an=5	darkgreen	
WFBAR1	waterfall	Bar	1.5 mg/kg	trt01an=6	darkturquoise	r3

Figure 2. Sample Waterfall Bar Metadata

The following diagram shows how the metadata for label, color and pattern are applied to the output RTF figure:

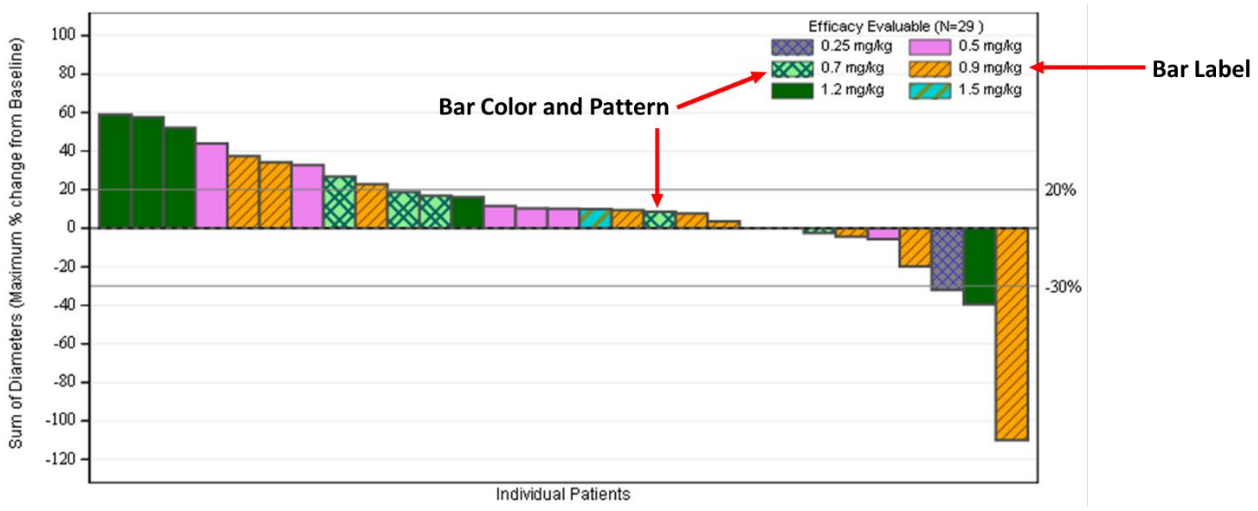
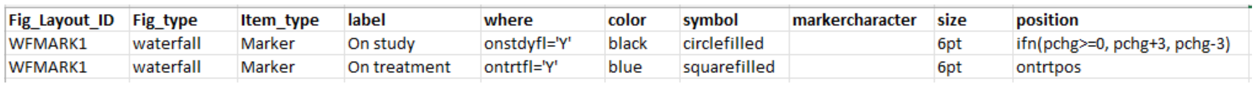


Figure 3. Bar in Waterfall Plot

Only rows that satisfy the WHERE statement are plotted for each row of metadata. To link to the metadata, we use the IN_BAR_DS macro parameter, e.g., IN_BAR_DS = WFBAR1.

METADATA FOR WATERFALL MARKERS

Markers may be defined using the standard SAS marker symbols or may be defined as character values. Here is sample metadata:



Fig_Layout_ID	Fig_type	Item_type	label	where	color	symbol	markercharacter	size	position
WFMARK1	waterfall	Marker	On study	onstdyfl='Y'	black	circlefilled		6pt	ifn(pchg>=0, pchg+3, pchg-3)
WFMARK1	waterfall	Marker	On treatment	ontrtfl='Y'	blue	squarefilled		6pt	ontrtpos

Figure 4. Sample Waterfall Marker Metadata

Marker metadata also includes POSITION metadata, which can be variable or a SAS statement acting on the input data set. This is used to define the vertical offset of the marker relative to the other data. To illustrate, for the Waterfall we display 2 markers in this example, and the position variable sets the location above or below the main analysis PCHG variable that is plotting the bar:

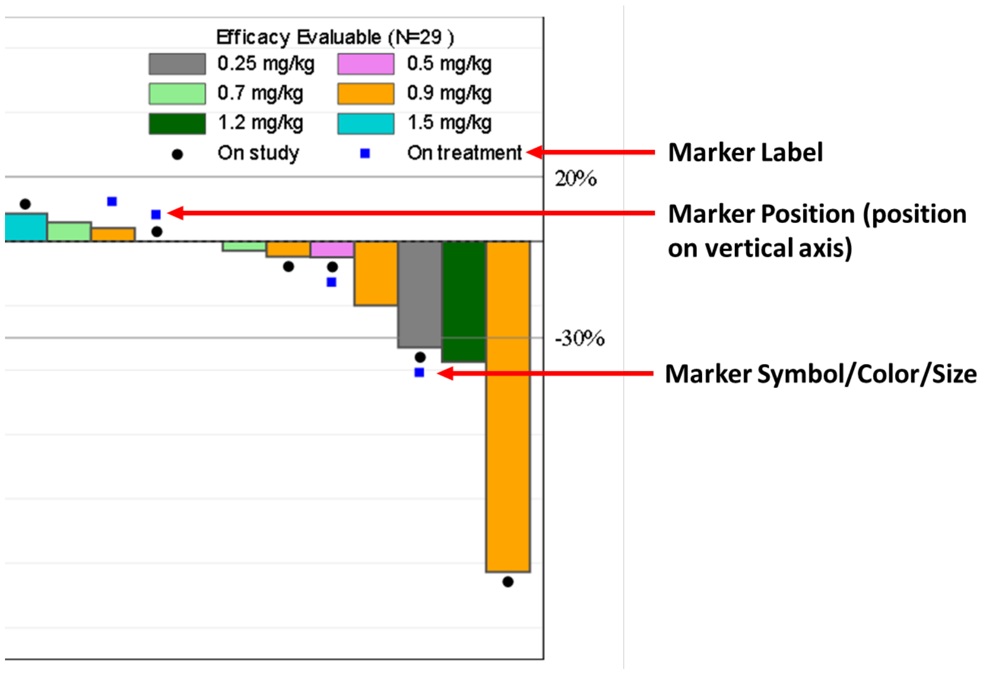
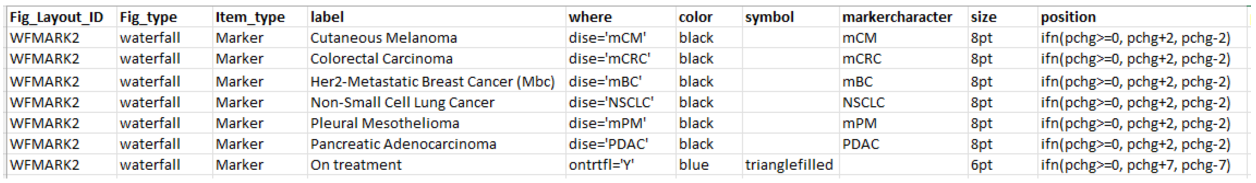


Figure 5. Marker Symbols in Waterfall Plot

The macro will only plot a marker symbol if the where statement for the row is true for the row of input data. In this example, the IFN function is used to plot the black on-study circle at position of percentage change (PCHG) +3 for non-negative bars and -3 for negative bars. This offsets the circle from the bar.

Markers may also be defined as a text value, using the markercharacter column instead of the symbol column:



Fig_Layout_ID	Fig_type	Item_type	label	where	color	symbol	markercharacter	size	position
WFMARK2	waterfall	Marker	Cutaneous Melanoma	dise='mCM'	black		mCM	8pt	ifn(pchg>=0, pchg+2, pchg-2)
WFMARK2	waterfall	Marker	Colorectal Carcinoma	dise='mCRC'	black		mCRC	8pt	ifn(pchg>=0, pchg+2, pchg-2)
WFMARK2	waterfall	Marker	Her2-Metastatic Breast Cancer (Mbc)	dise='mBC'	black		mBC	8pt	ifn(pchg>=0, pchg+2, pchg-2)
WFMARK2	waterfall	Marker	Non-Small Cell Lung Cancer	dise='NSCLC'	black		NSCLC	8pt	ifn(pchg>=0, pchg+2, pchg-2)
WFMARK2	waterfall	Marker	Pleural Mesothelioma	dise='mPM'	black		mPM	8pt	ifn(pchg>=0, pchg+2, pchg-2)
WFMARK2	waterfall	Marker	Pancreatic Adenocarcinoma	dise='PDAC'	black		PDAC	8pt	ifn(pchg>=0, pchg+2, pchg-2)
WFMARK2	waterfall	Marker	On treatment	ontrtfl='Y'	blue	trianglefilled		6pt	ifn(pchg>=0, pchg+7, pchg-7)

Figure 6. Sample Waterfall Markercharacter Metadata

Figure 7 shows how the marker character values like mCM and mCRC, which indicate the disease type, are plotted at the position +2 or -2 above/below the bars, and the on-treatment marker symbol (trianglefilled) is plotted at the position +7 or -7 above/below the bars.

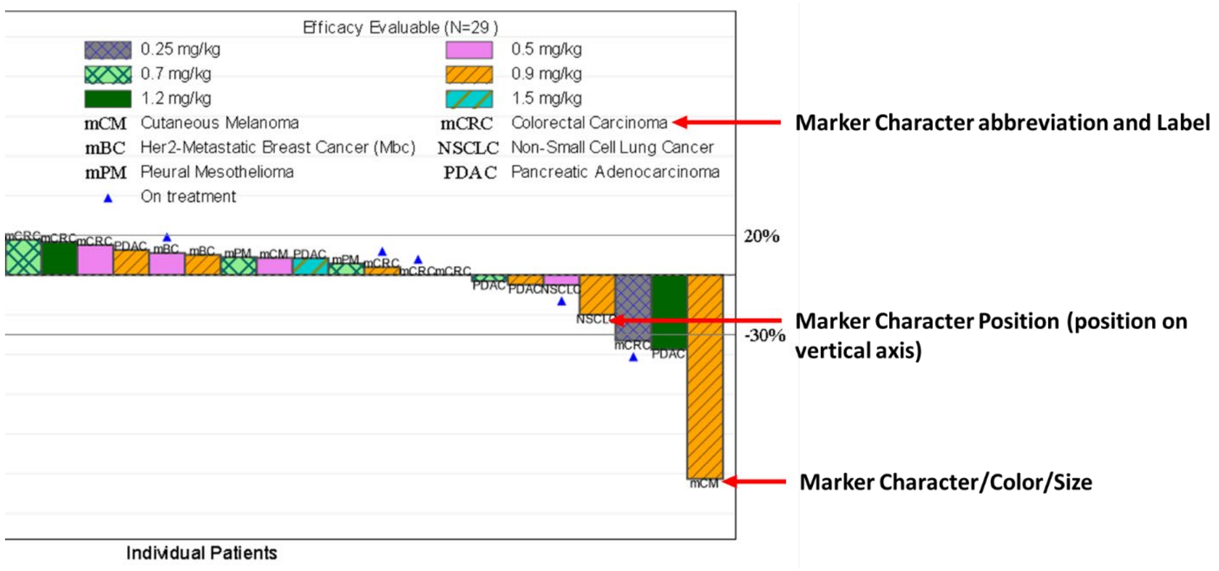


Figure 7. Marker Characters in Waterfall Plot

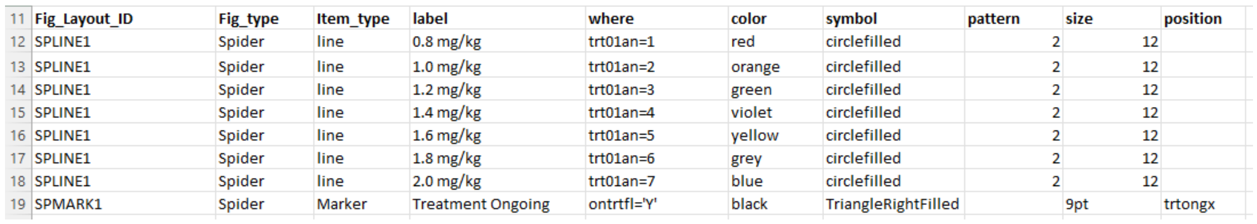
METADATA FOR THE SPIDER PLOT MACRO

A spider plot typically displays the percent change from baseline for a given parameter. This plot is usually generated for percent change in SPD (sum of the products of diameters) over time. The macro also supports creation of spider type figures for other parameters. The x-axis is the time since first dose in months or weeks. Each line in the graph represents one subject. In the example below, the different colors for the lines are used to represent dose levels. The black triangle in the graph represents the treatment ongoing marker.

In spider plots, each line represents one subject and corresponds to one row in the in_ds data set. Users may define any number of line types with different colors or patterns and may define any type of line markers with different symbols or characters. The direction and height of the line depend on the result variable in the in_ds data set.

METADATA FOR SPIDER PLOT LINES

Metadata for lines has the following structure. Each row of the metadata is used to define line attributes for the rows of the input data:



11	Fig_Layout_ID	Fig_type	Item_type	label	where	color	symbol	pattern	size	position
12	SPLINE1	Spider	line	0.8 mg/kg	trt01an=1	red	circlefilled	2	12	
13	SPLINE1	Spider	line	1.0 mg/kg	trt01an=2	orange	circlefilled	2	12	
14	SPLINE1	Spider	line	1.2 mg/kg	trt01an=3	green	circlefilled	2	12	
15	SPLINE1	Spider	line	1.4 mg/kg	trt01an=4	violet	circlefilled	2	12	
16	SPLINE1	Spider	line	1.6 mg/kg	trt01an=5	yellow	circlefilled	2	12	
17	SPLINE1	Spider	line	1.8 mg/kg	trt01an=6	grey	circlefilled	2	12	
18	SPLINE1	Spider	line	2.0 mg/kg	trt01an=7	blue	circlefilled	2	12	
19	SPMARK1	Spider	Marker	Treatment Ongoing	ontrtfl='Y'	black	TriangleRightFilled		9pt	trtongx

Figure 8. Sample Spider Plot Line Metadata

Figure 9 shows how the metadata for label, color and pattern are applied to the output RTF figure:

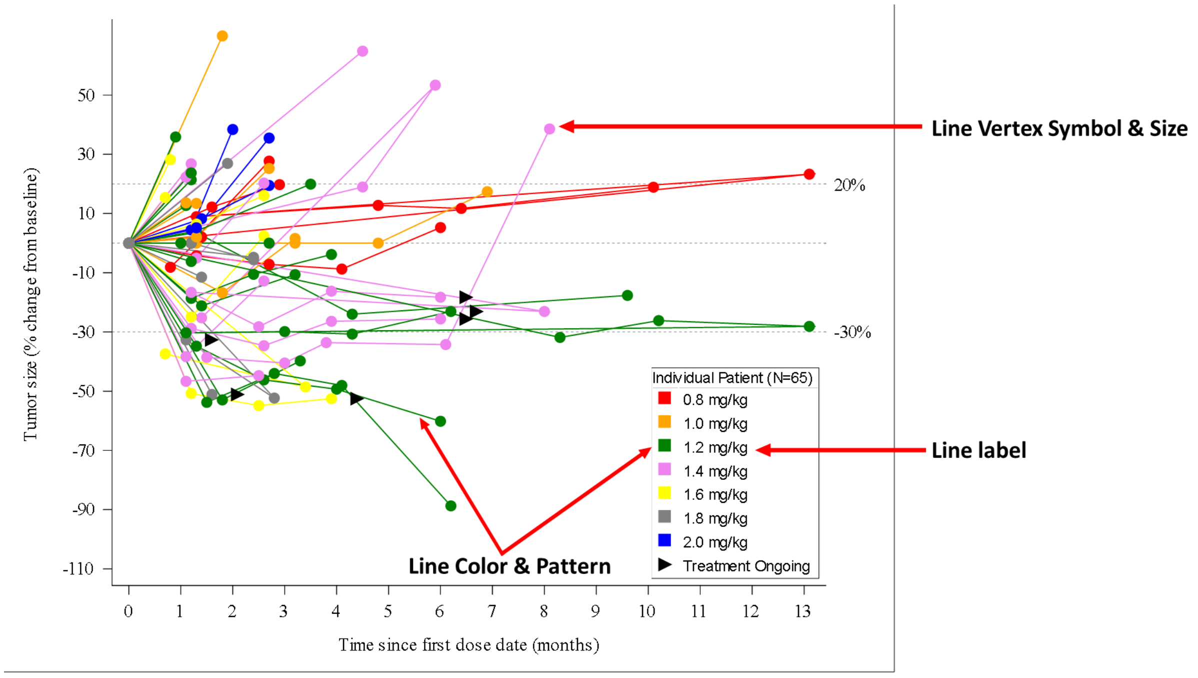


Figure 9. Lines in Spider Plot

METADATA FOR THE SWIMPLOT MACRO

The swimplot macro generates swimmer plots suitable for studies that analyze tumor response. A swimmer plot displays the response to the study drug over time in months. Each horizontal bar represents one subject in the study. The markers on the graph define trial milestones and the bar lengths represent duration of time.

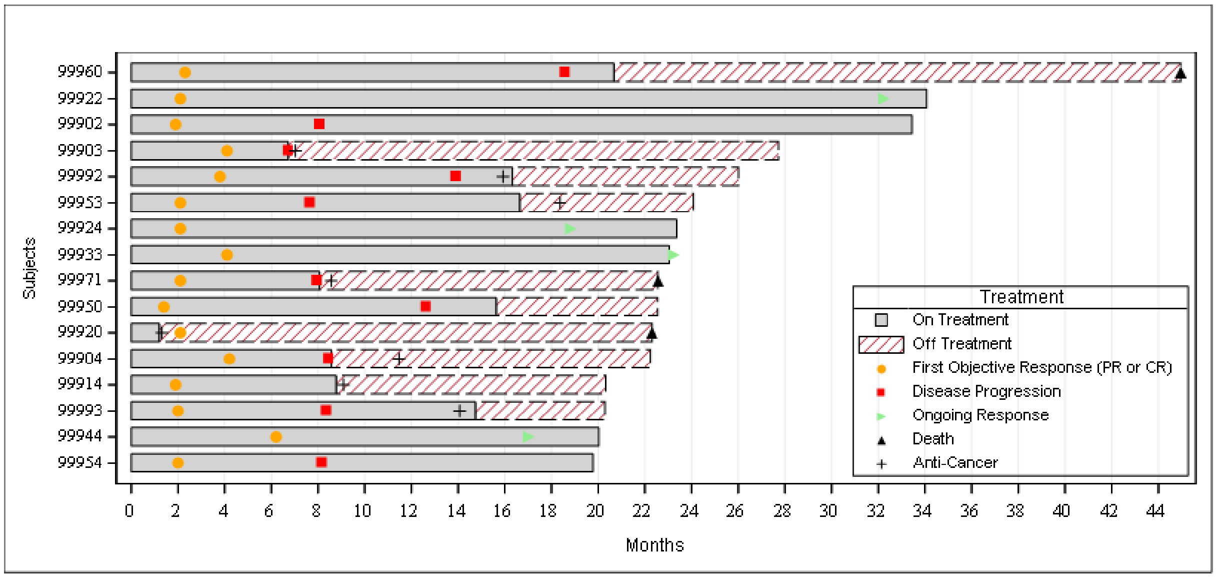
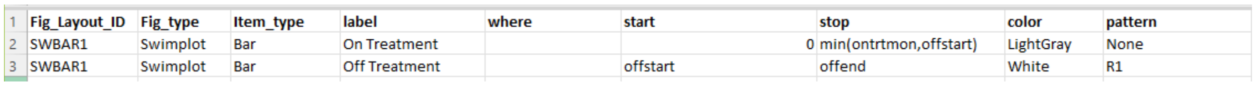


Figure 10. Swimmer Plot

METADATA FOR SWIMPLOT BARS

The swimmer plot contains horizontal bars for each subject. Each bar has a start and a stop position along the x-axis (in this example it is in months). The definition of start and stop position is provided in the bar metadata.



1	Fig_Layout_ID	Fig_type	Item_type	label	where	start	stop	color	pattern
2	SWBAR1	Swimplot	Bar	On Treatment		0	min(ontrtmon,offstart)	LightGray	None
3	SWBAR1	Swimplot	Bar	Off Treatment		offstart	offend	White	R1

Figure 11. Sample Swimplot Bar Metadata

The start and stop columns are SAS expressions that act on the in_ds data set. Start/end may be a numeric value like 0, a variable like offstart and offend from the &in_ds data set, or a more complex SAS expression like min(ontrtmon, offstart) that will be evaluated and plotted.

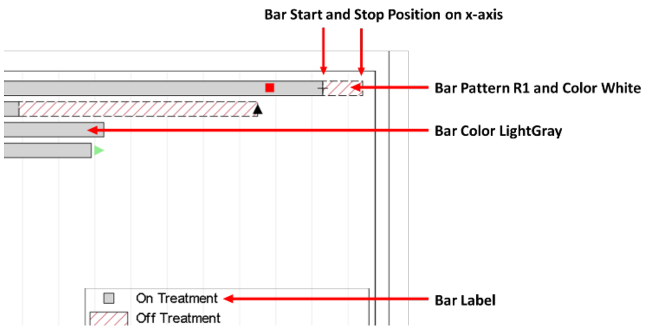
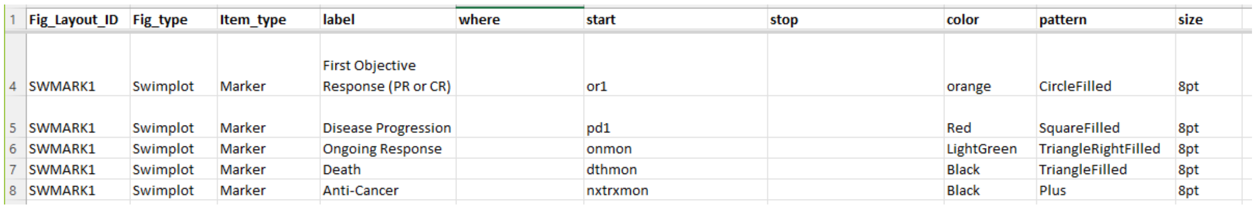


Figure 12. Bars in Swimplot

METADATA FOR SWIMPLOT MARKERS

The metadata for swimplot markers is very similar to waterfall markers. The main difference is that swimplot markers have a “start” position that is aligned along the x-axis (time).



1	Fig_Layout_ID	Fig_type	Item_type	label	where	start	stop	color	pattern	size
4	SWMARK1	Swimplot	Marker	First Objective Response (PR or CR)		or1		orange	CircleFilled	8pt
5	SWMARK1	Swimplot	Marker	Disease Progression		pd1		Red	SquareFilled	8pt
6	SWMARK1	Swimplot	Marker	Ongoing Response		onmon		LightGreen	TriangleRightFilled	8pt
7	SWMARK1	Swimplot	Marker	Death		dthmon		Black	TriangleFilled	8pt
8	SWMARK1	Swimplot	Marker	Anti-Cancer		nxtrxmon		Black	Plus	8pt

Figure 13. Sample Swimplot Maker Metadata

The start column in the metadata is used to assign the position of the marker along the x-axis. In Figure 14 disease progression has start=pd1, where PD1 is a variable in the macro input data set (the data to be plotted).

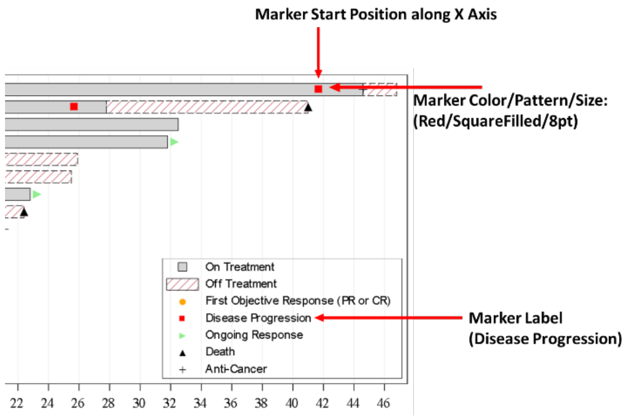


Figure 14. Markers in Swimplot

MACRO TECHNIQUES USING THE METADATA

READING IN EXCEL OR SAS DATA SET METADATA

Figure metadata may be stored either in an Excel spreadsheet or stored in a simple SAS data set created in the calling program. For example, here is some bar metadata for the waterfall plot:

Excel spreadsheet format:

Fig_Layout_ID	Fig_type	Item_type	label	where	color	pattern
WFBAR1	waterfall	Bar	0.25 mg/kg	trt01an=1	gray	x1
WFBAR1	waterfall	Bar	0.5 mg/kg	trt01an=2	violet	
WFBAR1	waterfall	Bar	0.7 mg/kg	trt01an=3	lightgreen	x5

User-defined SAS data set:

```
data bar_meta;
  format label where color pattern $200.;
  label="0.25 mg/kg"; where = "trt01an=1"; color="gray";       pattern="x1"; output;
  label="0.5 mg/kg";  where = "trt01an=2"; color="violet";     pattern="";   output;
  label="0.7 mg/kg";  where = "trt01an=3"; color="lightgreen"; pattern="x5"; output;
run;
```

A "get metadata" macro reads in the Excel or SAS data set metadata and converts it into individual data sets for each figure element (e.g., bar_data or marker_data). The macro also helps check the metadata and reports any data issues. For example, the variable used in a WHERE statement must exist in data set IN_DS, and the color specified in COLOR will be checked against valid SAS colors.

CONVERTING THE METADATA TO MACRO VARIABLES

After reading in figure metadata, macro variables are created for the attributes of figure elements. In the sample code below, multiple macro parameters are generated corresponding to bar label, color, pattern and where statement.

```
      %if &spi_bar_nobs > 0 %then %do;
           data _null_;
             set spifwf_bar_data1;
             if upcase(pattern) in ("", "NONE") then pattern = "empty";
             call symput("spi_bar_label"||strip(put(_n_,8.)),strip(label));
             call symput("spi_bar_where"||strip(put(_n_,8.)),strip(where));
             call symput("spi_bar_color"||strip(put(_n_,8.)),strip(color));
             call symput("spi_bar_pattern"||strip(put(_n_,8.)),strip(pattern));
           run;
      %end;
```

For example, the highlighted code within the call symput statement assigns macro variables spi_bar_label1, spi_bar_label2, spi_bar_label3 to the value of the metadata label column for rows 1, 2, 3:

Macro Variable	**Value**
spi_bar_label1	0.25 mg/kg
spi_bar_label2	0.5 mg/kg
spi_bar_label3	0.7 mg/kg

We also obtain the number of rows of metadata and store this in a macro variable spi_bar_nobs. The overall programming strategy is to use macro %do loops to loop through and operate on the macro variables generated here.

USING METADATA MACRO VARIABLES TO PREPARE THE INPUT DATA

Based on input data set IN_DS, metadata macro variables are used to create variables which will be used for plotting. For example, bargroup and barlabel are created when the where condition is satisfied:

```
data fullset;
  set &in_ds;
. . . ;
    %do spi_i = 1 %to &spi_bar_nobs;
        %if %length(&&spi_bar_where&spi_i.) > 0 %then %do;
            if &&spi_bar_where&spi_i then do;
                spi_bargroup = &spi_i;
                spi_barlabel = "&&spi_bar_label&spi_i.";
            end;
        %end;
    %end;
. . . ;
run;
```

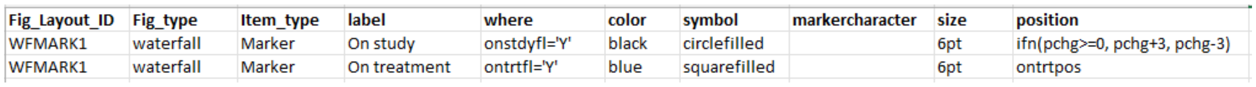
When the data step is evaluated in the macro, SAS will evaluate as follows:

```
data fullset;
  set &in_ds;
. . . ;

       if trt01an=1 then do;
           spi_bargroup = 1;
           spi_barlabel = ”0.25 mg/kg”;
       end;
       if trt01an=2 then do;
           spi_bargroup = 2;
           spi_barlabel = ”0.5 mg/kg”;
       end;
       if trt01an=3 then do;
           spi_bargroup = 3;
           spi_barlabel = ”0.7 mg/kg”;
       end;

. . . ;
run;
```

For markers, suppose we are using the following metadata to define 2 marker symbols:



Fig_Layout_ID	Fig_type	Item_type	label	where	color	symbol	markercharacter	size	position
WFMARK1	waterfall	Marker	On study	onstdyfl='Y'	black	circlefilled		6pt	ifn(pchg>=0, pchg+3, pchg-3)
WFMARK1	waterfall	Marker	On treatment	ontrtfl='Y'	blue	squarefilled		6pt	ontrtpos

To plot the marker symbols, the following macro code is used to loop through the marker symbols:

```
data fullset;
  set &in_ds end=end;
  . . . ;
    %do spi_i = 1 %to &spi_marker_nobs;
         %if %length(&&spi_marker_where&spi_i.) > 0 %then %do;
            retain spi_found_marker&spi_i;
            if &&spi_marker_where&spi_i. then do;
                z&spi_i.=&&spi_marker_position&spi_i.;
                if z&spi_i ne . then spi_found_marker&spi_i = 1;
            end;
            if end and spi_found_marker&spi_i = 1
              then call symput("spi_marker_found&spi_i","Y");

         %end;
    %end;
 . . .;
run;
```

Upon macro execution, the macro %do loop code will be evaluated as follows:

```
data fullset;
  set &in_ds end=end;
  . . . ;
            if onstdyfl="Y" then do;
                retain spi_found_marker1;
                z1=ifn(pchg>=0,pchg+3,pchg-3);
                if z1 ne . then spi_found_marker1 = 1;
                if end and spi_found_marker1 = 1 then call symput("spi_marker_found1","Y");
            end;
            if ontrtfl="Y" then do;
                retain spi_found_marker2;
                z2=ontrtpos;
                if z2 ne . then spi_found_marker2 = 1;
                if end and spi_found_marker2 = 1 then call symput("spi_marker_found2","Y");
            end;

. . .;
run;
```

The above code creates variables z1, z2, etc. that will be used within a scatter plot statement for the position of the marker symbols. Using call symput, a set of macro variables &spi_marker_found1, &spi_marker_found2, etc are created to check if any marker values are non-missing. Only markers that are non-missing in the figure are included in the legend.

USING METADATA MACRO VARIABLES TO DEFINE STYLE ELEMENTS

Within the macro code we define style elements corresponding to attributes like bar colors and patterns.

```
    proc template;
      define style wfstyle;
        parent=rtftnr10;
        style graphfonts from graphfonts /
          'GraphDataFont' =("Arial",8pt)
          'GraphDataValue'=("Arial",8pt)
          'GraphLabelFont'=('Arial',9pt)
          'GraphTitleFont'=('Arial',8pt)
          'GraphFootnoteFont'=('Arial',10pt);

            %do spi_i = 1 %to &spi_bar_nobs;
                style GraphData&spi_i. from GraphData&spi_i. / color=&&spi_bar_color&spi_i.
fillpattern="&&spi_bar_pattern&spi_i.";
            %end;
      end;
    run;
```

Here the %do loop is creating style GraphData elements that will be used later for plotting using the barchart statement.

USING METADATA MACRO VARIABLES TO DEFINE A STATGRAPH

For plotting the figure we use PROC SGRENDER along with the PROC TEMPLATE statgraph element.

```
proc template;
  define statgraph wf;
  . . .;
  end;
run;
proc sgrender data=final template=wf objectlabel="&hidetitle.";
run;
```

All of the figure drawing instructions are contained within the statgraph definition.

To draw the bars, we use the barchart statement within the statgraph definition. Note that the index=spi_bargroup option is being used to format based on the GraphData1, GraphData2, etc. styles defined above. For example, when spi_bargroup=3 in the input data, the GraphData3 style will be used for the bars.

```
barchart y=&result_var. x=x /
         display=(fillpattern fill outline ) barwidth=&bar_width. index=spi_bargroup
         outlineattrs=(thickness=2px color=&bar_border_color.) group=spi_bargroup
         grouporder=ascending name='trt';
```

To draw marker symbols we use a scatterplot statement within the statgraph definition.

Here we are setting y=z&spi_i. which refers to the “z” variables we defined earlier for marker vertical positions. It also shows how we use the spi_marker_found&spi_i macro variables that we defined earlier.

```
%do spi_i = 1 %to &spi_marker_nobs;
  %if &&spi_marker_found&spi_i eq Y %then %do;
     scatterplot x=x y=z&spi_i. /
         markerattrs= (size  = &&spi_marker_size&spi_i.
                       color = &&spi_marker_color&spi_i.
                       symbol= &&spi_marker_symbol&spi_i.);
   %end;
%end;
```

To draw legend items for markers within the legend, we use the legenditem statement within the statgraph definition:

```
%do spi_i = 1 %to &spi_marker_nobs;
   %if &&spi_marker_symbol&spi_i.^= %then %do;
       legenditem
           type = marker
           name = "marker&spi_i" /
                  markerattrs=(color=&&spi_marker_color&spi_i.
                  symbol=&&spi_marker_symbol&spi_i.) label="&&spi_marker_label&spi_i.";
   %end;
%end;
```

And then we define a custom discretelegend statement to combine the marker and bar entries into a single legend:

```
discretelegend 'trt'
     %do spi_i = 1 %to &spi_marker_nobs;
         %if &&spi_marker_found&spi_i eq Y %then %do;
                "marker&spi_i"
         %end;
     %end;
     /
     title="&legend_title."
     titleattrs=(family="Arial" size=&legend_fontsize.)
     titleborder=true across=2 sortorder=auto
     autoalign=(&legend_location.) location=inside
     order=rowmajor border=false valueattrs=(family="Arial" size=&legend_fontsize.);
```

CONCLUSION

In this paper we showed how we use metadata as an input for a set of figure macros that are intended to accelerate the production of clinical trial figures. The metadata includes columns like “color”, “pattern”, “symbol”, “size” and “label” that are used to define the visual attributes of the figure element bars, lines and markers. The metadata also includes a “where” column that is used to select rows of input data, and “position”, “start” and “stop” columns to define the positions of figure elements within the graph.

With this common approach to swimmer's, waterfall, and spider plot figures, we standardized the figure generation within the macros using PROC TEMPLATE and PROC SGRENDER. We described the common SAS macro programming techniques that convert metadata to macro variables and how to use macro DO loops to process the metadata. The macro creates a customized statgraph template to generate the figure.

While programmers are still required to prepare the input data in the programs, use of these macros will allow teams to both standardize their figure outputs using common metadata across studies and be responsive to the specific needs of individual trials through the ability to easily add any number of bars, lines, and markers to the figures.

CONTACT INFORMATION

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

Jeremy Gratt, Modular Informatics, LLC, jeremy.gratt@m-informatics.com

Qiuhong Jia, Seagen Inc., qji@seagen.com

Girish Kankipati, Seagen Inc., gkankipati@seagen.com