PharmaSUG 2024 - Paper HT- 197

Building Complex Graphics from Simple Plot Types

Dan Heath, SAS Institute

ABSTRACT

There are many types of graphics, such as survival plots, adverse event plots, and forest plots, used to report clinical data. However, many of these graphics might not be available as a simple request. Even if they are available, there can be many variations on these graphics, and the reporting requirements can be different from one company to another.

In this hands-on workshop, I want to show you how to construct these types of graphics by combining simple plot types, sometimes in very creative ways. We will discuss how to think about a graph as the sum of its parts. This will empower you to create these types of graphics in whatever form you need.

INTRODUCTION

Effective graphics are an important part of reporting clinical results and information. The content within plots can range from simple to complex. However, if you look at a complex plot as the "sum of its parts" instead of as single entity, the process for creating these plots can become a lot clearer.

In this paper, I will discuss techniques for analyzing complex plots to break them down into simpler plot types. Then, how you can assemble those types into one cohesive plot. Along the way, you will discover creative ways to use plot types beyond their originally intended usage.

Almost all the examples in this paper are created using the SGPLOT procedure that is part of the SAS® ODS Graphics system; however, these techniques can be applied to the SGPANEL procedure as well, when classification panels are needed to convey the needed information. The discussion regarding simple plot types will be referencing plot types available in these two procedures.

WATERFALL PLOT VARIATIONS

The following examples of waterfall plots will illustrate the importance of proper overlay ordering. In Figure 1, the plot is constructed using a bar chart (VBARPARM) and two reference lines (REFLINE) to show clinical significance. The VBARPARM statement is a vertical bar chart with no summarization. The data contains one tumor response per patient ID, so no summarization is required. The VBARPARM statement has more overlay flexibility than a VBAR statement, so I used it instead of the VBAR statement for all the following examples. Also, both reference lines are combined into one statement, as both lines have the same visual attributes:

```
proc sort data=TumorSize out=TumorSizeDesc;
  by descending change;
run;

title 'Change in Tumor Size';
title2 'ITT Population';
proc sgplot data=TumorSizeDesc nowall noborder;
  format label $resp.;
  vbarparm category=cid  response=change / group=label dataskin=pressed;
  refline 20 -30 / lineattrs=(color=black pattern=shortdash);
  xaxis display=none;
  yaxis values=(60 to -100 by -20);
  keylegend / title='';
run;
```

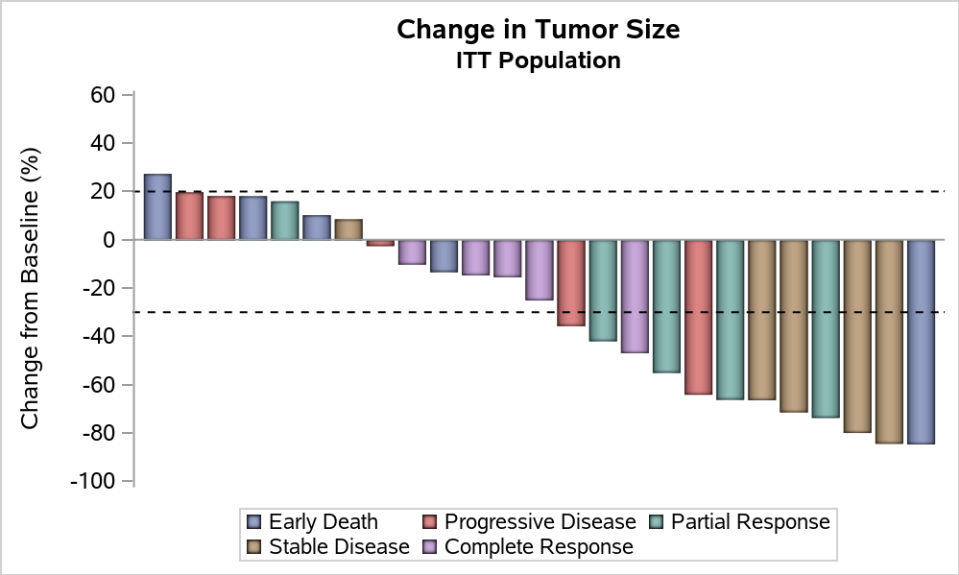

Figure 1. Waterfall plot with reference lines

The order of the bars is important for a waterfall plot. The bars should be sorted from largest tumor increase to largest tumor decrease. With a VBAR statement, this would be done by setting the CATEGORYORDER option to RESPDESC (response descending). However, this option is not currently available on the VBARPARM statement, so the data must be sorted before using it in the procedure:

The position of the REFLINE statement after the VBARPARM statement is significant. With this statement specified after the bar chart, the reference lines are drawn on top of the bars. Otherwise, the reference lines would be partially obscured by the bar chart. Plot statements are drawn the order they are specified. This rule does not apply to other procedure statements, such as axis and legend statements.

To illustrate the rule further, I used a band plot to highlight the region between the two reference points. However, I put the BAND statement in the same location as the REFLINE statement, and you can see that the band plot overwrites the bars (Figure 2). It is worth noting that there are situations where the TRANSPARENCY option can be used with filled regions so that you can see plot features that would have otherwise been obscured, such as grid lines and intersecting filled areas. But, as a rule, you will want to construct your plots in a way that plots with filled areas are drawn first, and line and marker-based plots are drawn last.

```
title 'Change in Tumor Size';
title2 'ITT Population';
proc sgplot data=TumorSizeDesc nowall noborder;
  format label $resp.;
  vbarparm category=cid  response=change / group=label dataskin=pressed
           name="waterfall";
  band  x=cid upper=20 lower=-30 ;
  xaxis display=none;
  yaxis values=(60 to -100 by -20);
  keylegend "waterfall" / title='';
run;
```

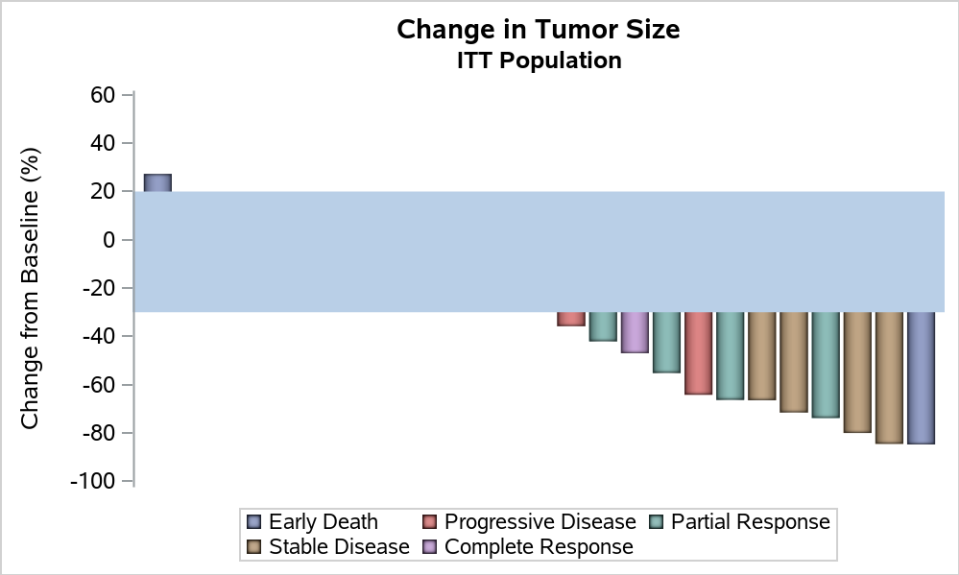

Figure 2. Band plot covers the waterfall bars

To fix the issue in Figure 2, simply swap the position of the two statements so that the band is drawn first (Figure 3):

```
title 'Change in Tumor Size';
title2 'ITT Population';
proc sgplot data=TumorSizeDesc nowall noborder;
  format label $resp.;
  band   x=cid upper=20 lower=-30 ;
  vbarparm category=cid  response=change / group=label dataskin=pressed
           name="waterfall";
  xaxis display=none;
  yaxis values=(60 to -100 by -20);
  keylegend "waterfall" / title='';
run;
```

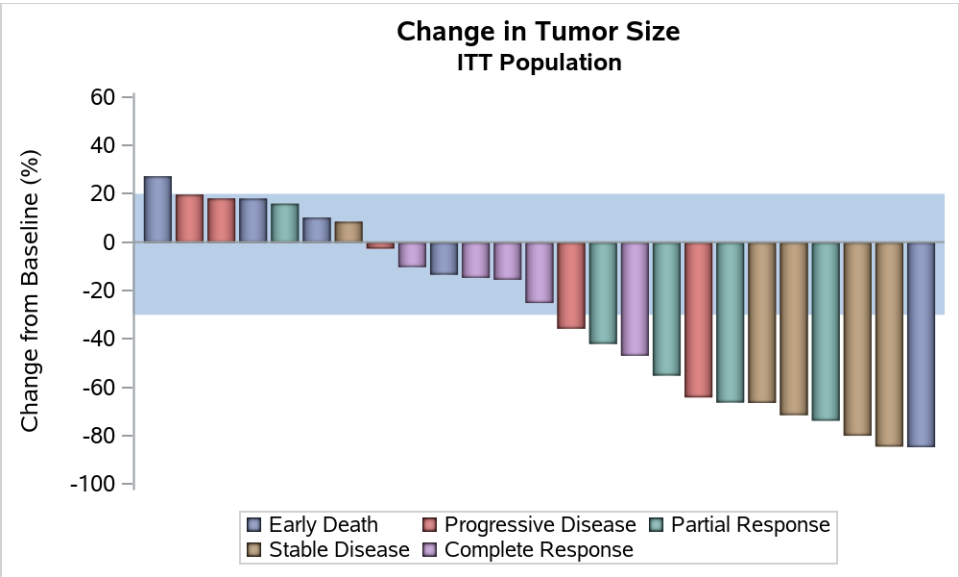

Figure 3. Waterfall plot with the band behind the bars

Band plots do not have to be filled. In fact, you can use a band plot to recreate the same reference line appearance we had in the original waterfall plot example by specifying the NOFILL and OUTLINE options (Figure 4):

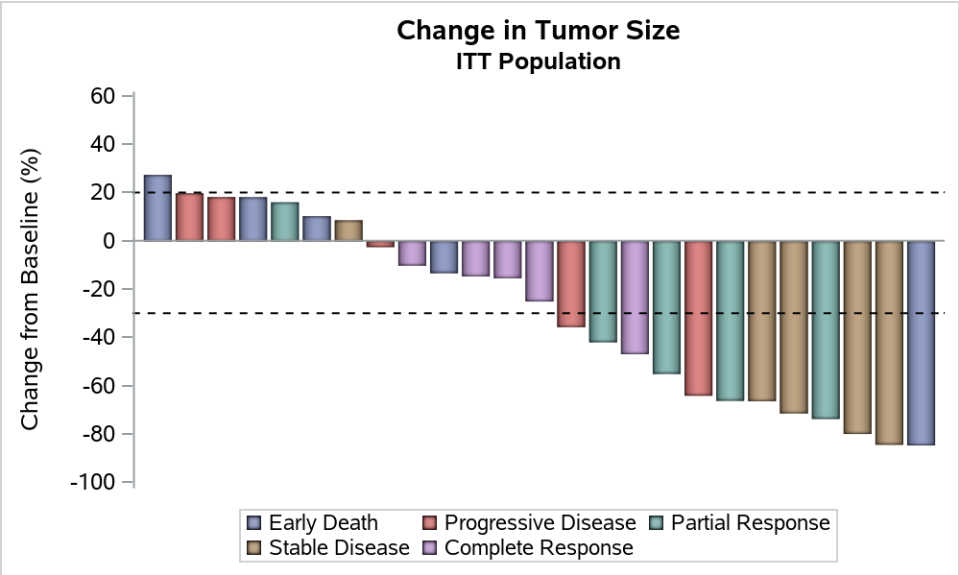

Figure 4. Waterfall plot using a band plot for reference lines

```
title 'Change in Tumor Size';
title2 'ITT Population';
proc sgplot data=TumorSizeDesc nowall noborder;
  format label $resp.;
  vbarparm category=cid  response=change / group=label dataskin=pressed
name="waterfall";
  band x=cid upper=20 lower=-30 / lineattrs=(color=black pattern=shortdash)
nofill outline;
  xaxis display=none;
  yaxis values=(60 to -100 by -20);
  keylegend "waterfall" / title='';
run;
```

While we are on the topic of band plots, another creative use for a band plot can be for the purpose of creating “area” plots, which show areas under a curve that are filled to a baseline value. This is done by setting the constant baseline value on either the UPPER or LOWER options and setting the curve variable on the other option. Figure 5 demonstrates this technique using our waterfall plot tumor data:

```
title 'Change in Tumor Size';
title2 'ITT Population';
proc sgplot data=TumorSizeDesc nowall noborder noautolegend;
  band x=cid upper=change lower=0 ;
  refline 20 -30 / lineattrs=(color=black pattern=shortdash);
  xaxis display=none;
  yaxis values=(60 to -100 by -20);
run;
```

Notice that the plot is drawn correctly, even though the “upper” values drop below the “lower” baseline value.

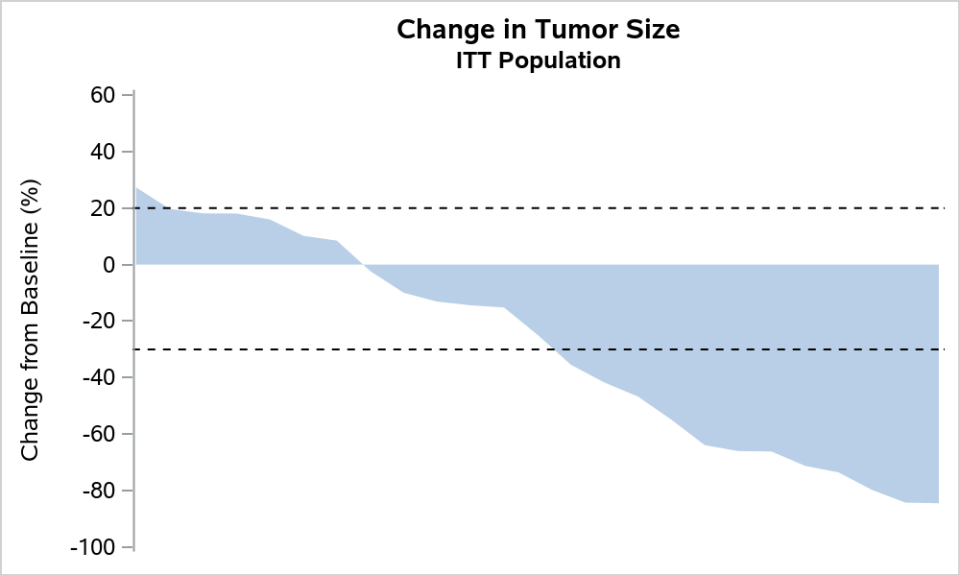

Figure 5. Area plot using a BAND statement

Coming back to our original waterfall plot, Figure 6 shows one way to add an additional level of classification to the plot. In this example, the treatment group column is assigned to the GROUP option and the response code column to the DATALABEL option to put the code values on the end of the bars. The INSET statement is used to create a “legend” for the code values.

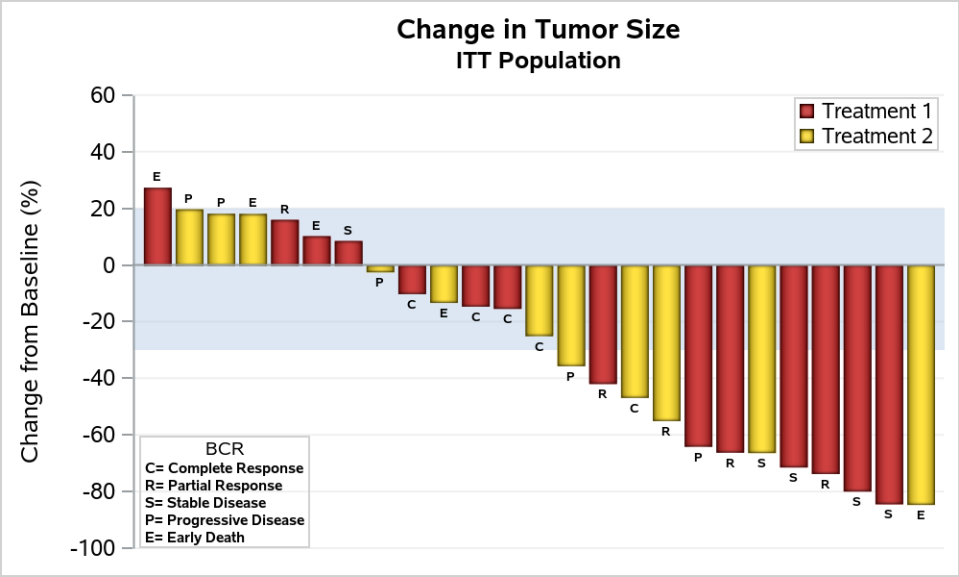

Figure 6. Waterfall plot showing both treatment group and response code

```
title 'Change in Tumor Size';
title2 'ITT Population';
proc sgplot data=TumorSizeDesc nowall noborder;
  styleattrs datacolors=(cxbf0000 gold) datacontrastcolors=(black)
             axisextent=data;
  band x=cid upper=20 lower=-30 / transparency=0.5;
  vbarparm category=cid  response=change / group=group datalabel=label
           datalabelattrs=(size=5 weight=bold) groupdisplay=cluster
           name="waterfall" dataskin=pressed;
  xaxis display=none;
```

```
  yaxis values=(60 to -100 by -20) grid gridattrs=(color=cxf0f0f0);
  inset "C= Complete Response" "R= Partial Response" "S= Stable Disease"
        "P= Progressive Disease" "E= Early Death" / title='BCR'
        position=bottomleft border textattrs=(size=6 weight=bold)
        titleattrs=(size=7);
  keylegend "waterfall" / title='' location=inside position=topright across=1
            border opaque;
run;
```

Figure 7 shows a slightly different take on this plot and, in the process, demonstrates another great utility plot – the text plot. The text plot is basically a scatter plot that uses text strings as markers. There are several options on the TEXT statement for controlling the appearance of the strings, as well as their positions.

In this example, I wanted to have all the codes written above the baseline if the bar is negative. There is not an option to give the DATALABEL option that behavior; therefore, I used a text plot to render the codes and removed the DATALABEL option. I used a small data step to generate the location of the text points, which is derived from the bar chart response values. If the bar value is less than zero, I just set the Y location to be zero:

```
data custom_label;
set TumorSizeDesc;
xlabel = cid;
ylabel = ifn(change < 0, 0, change);
run;
```

In the procedure, I added the TEXT statement to the previous example using the new columns to place the text. Notice the POSITION option on the TEXT statement. By default, the data point is in the center of the text. For this case, however, I needed the text to be on **top** of the point, so that the bars and baseline do not cut into the text: I also used the same text attributes that were originally used in the DATALABELATTRS option to give the text the same look as the original labels.

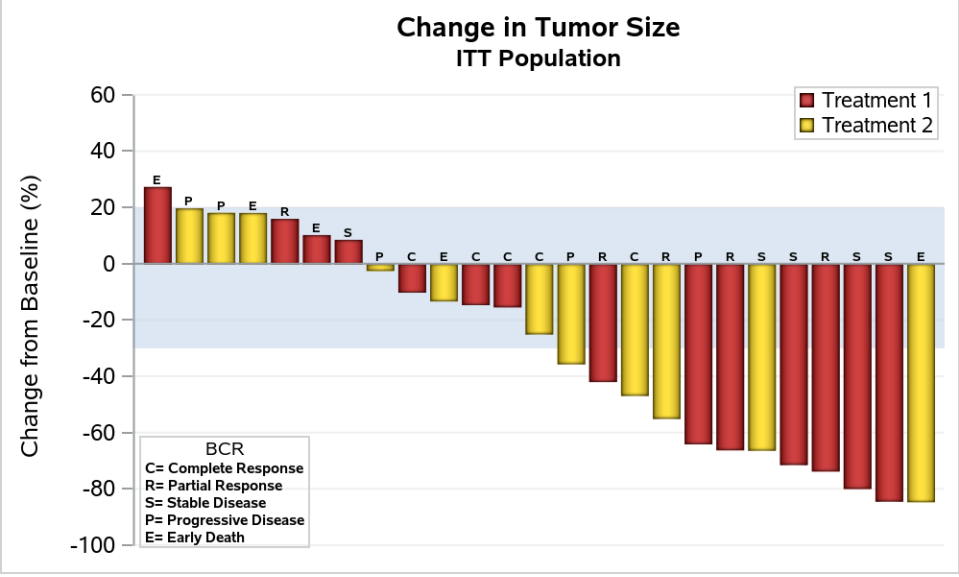

Figure 7. Waterfall plot with text plot labeling

```
title 'Change in Tumor Size';
title2 'ITT Population';
proc sgplot data=custom_label nowall noborder;
  styleattrs datacolors=(cxbf0000 gold) datacontrastcolors=(black)
             axisextent=data;
  band x=cid upper=20 lower=-30 / transparency=0.5;
```

```
  vbarparm category=cid response=change / group=group groupdisplay=cluster
           name="waterfall" dataskin=pressed;
  text x=xlabel y=ylabel text=label / textattrs=(size=5 weight=bold)
       position=top;
  xaxis display=none;
  yaxis values=(60 to -100 by -20) grid gridattrs=(color=cxf0f0f0);
  inset "C= Complete Response" "R= Partial Response" "S= Stable Disease"
        "P= Progressive Disease" "E= Early Death" / title='BCR'
        position=bottomleft border textattrs=(size=6 weight=bold)
        titleattrs=(size=7);
  keylegend "waterfall" / title='' location=inside position=topright across=1
            border opaque;
run;
```

Another alternative for displaying the additional classifier would be to use a classification panel, which you can do using the SGPANEL procedure. The same order consideration for plot statements also applies to this procedure. In Figure 8, I left the response code column on the GROUP option, but I put the treatment group column on the PANELBY statement. This creates a cell in the panel for each treatment group, and each cell contains only the patients in that treatment group:

By default, all column axes have a uniform range, and all row axes have a uniform range. In this case, however, I needed the column axes to be independently scaled, as each patient can be in only one treatment group. To do this, I used the UNISCALE option to set the ROW axes to uniform, making the column axes to be independent.

Since there are a different number of bars in each cell, I needed to have the cells automatically adjust their widths so that all bars have the same width. The PROPORTIONAL option will do this for you. I also used the NOVARNAME option to have only the group value displayed in the header.

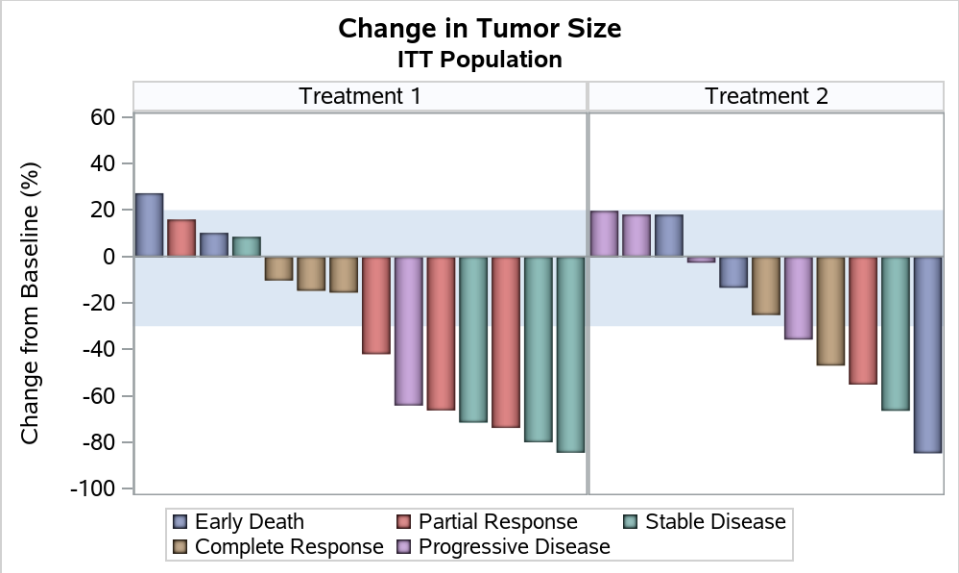

Figure 8.. Waterfall plot in a classification panel

```
title 'Change in Tumor Size';
title2 'ITT Population';
proc sgpanel data=TumorSizeDesc;
  format label $resp.;
  panelby group / uniscale=row novarname proportional;
  band x=cid upper=20 lower=-30 / transparency=0.5;
  vbarparm category=cid  response=change / group=label dataskin=pressed
           name="waterfall";
```

```
  colaxis display=none;
  rowaxis values=(60 to -100 by -20);
  keylegend "waterfall" / title='';
run;
```

SURVIVAL PLOT CONSTRUCTION

Figure 9 shows an example of a survival plot with Kaplan-Meier curves and an at-risk table. If we break this plot down to simple types, it consists of step plots, scatter plots, and X axis tables. The number of each plot type needed depends on the number of stratum and how your data is organized. For this example, the data is grouped by stratum; therefore, only one step plot is needed for the Kaplan-Meier curves. one scatter plot for the censored observations, and one axis table to represent the patients at-risk in each stratum:

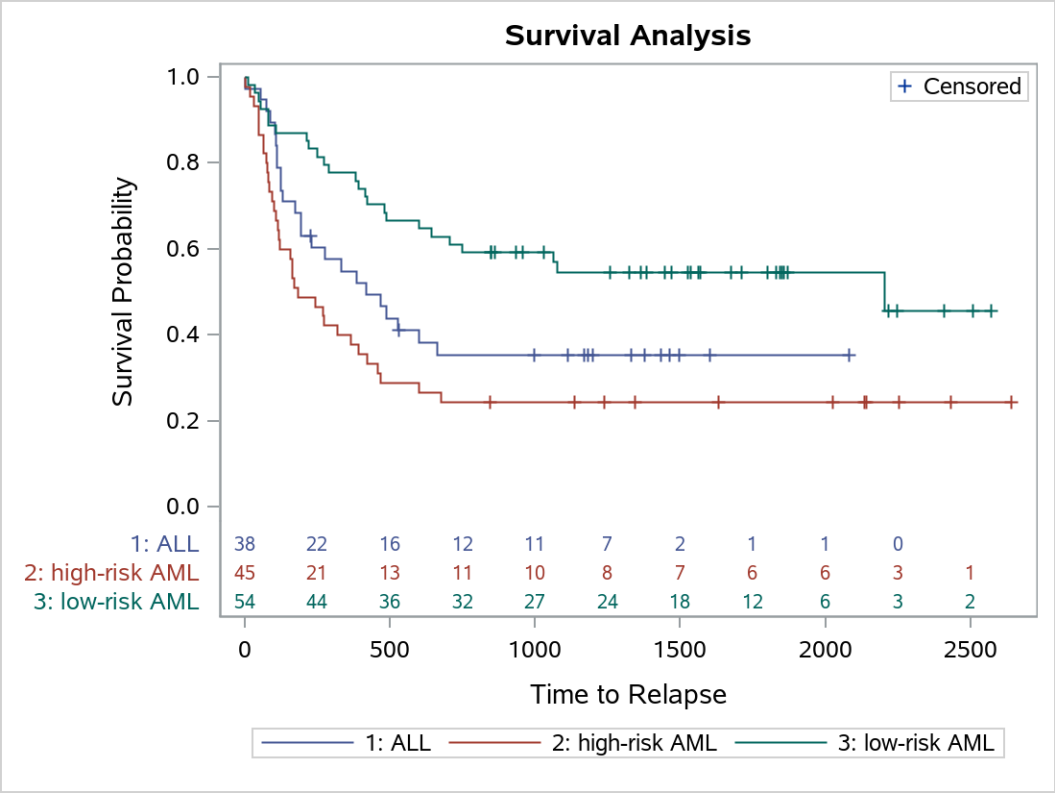

Figure 9. Survival plot with an at-risk table

```
title "Survival Analysis";
proc sgplot data=OutSurvival;
    step x=time y=survival / group=stratum name='s';
    scatter x=time y=censored / group=stratum markerattrs=(symbol=plus);
    xaxistable atrisk / x=tatrisk class=stratum location=inside
              colorgroup=stratum;
    legenditem type=marker name='c' label='Censored'/
              markerattrs=GraphDataDefault(symbol=plus);
    yaxis min=0;
    keylegend 'c' / location=inside position=topright;
    keylegend 's' / outerpad=10;
run;
```

This plot contains two legends – one for the stratum and one for the censored observations. The stratum legend clearly shows each group, so it would be redundant to show the same group values in the

censored legend. Instead of associating the legend with the scatter plot, which would show the group values, the example uses a LEGENDITEM and associates it with the legend instead. A legend item contributes to the content of a legend as if it were a plot, but it has no other impact on the plot visual.

One other common request for this plot is to have the at-risk table be outside of the plot area. With the SGPLOT procedure, this can be done simply by specifying LOCATION=OUTSIDE on the axis table statement (Figure 10):

```
title "Survival Analysis";
proc sgplot data=OutSurvival;
    step x=time y=survival / group=stratum name='s';
    scatter x=time y=censored / group=stratum markerattrs=(symbol=plus);
    xaxistable atrisk / x=tatrisk class=stratum
                colorgroup=stratum location=outside;
    legenditem type=marker name='c' /
                markerattrs=GraphDataDefault(symbol=plus)
                label='Censored';
    yaxis min=0;
    keylegend 'c' / location=inside position=topright;
    keylegend 's' / outerpad=10;
run;
```

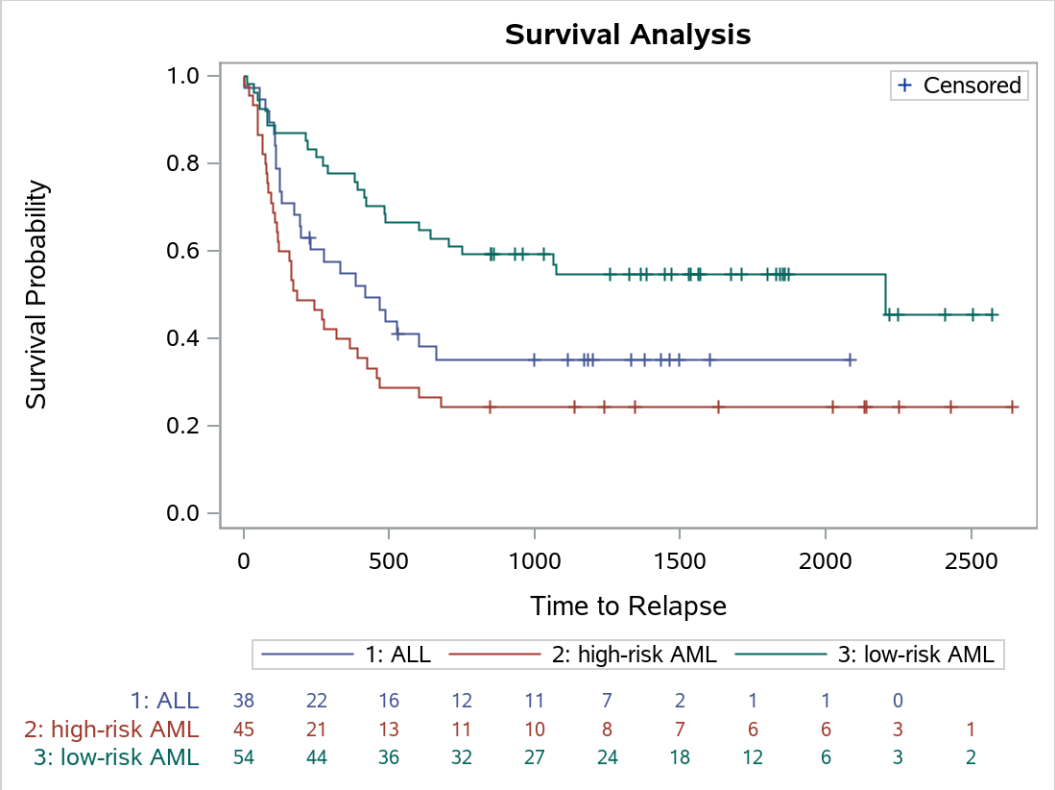

Figure 10. Survival plot with an at-risk table

FOREST PLOT CONSTRUCTION

Forest plots are used to report study results, and can be constructed in various ways, depending on the information that needs to be included. These plots typically contain a list of studies, an odds-ratio plot and additional axis-aligned information.

Forest plots are horizontally oriented to accommodate all the information they show. You will want to start the construction by turning off the display of the Y axis and use axis tables for the display of all axis-aligned information, including the studies (see Figure 11):

```
title "Impact of Treatment on Mortality by Study";
title2 h=8pt 'Odds Ratio and 95% CL';
proc sgplot data=forest noautolegend nocycleattrs;
  styleattrs datasymbols=(squarefilled diamondfilled);
  scatter y=study x=or / xerrorupper=ucl xerrorlower=lcl group=grp
         noerrorcaps;
  yaxistable study / y=study location=inside position=left;
  yaxistable or lcl ucl wt / y=study location=inside position=right
         nomissingchar;
  refline 0.01 1 100  / axis=x noclip;
  refline 0.1 10 / axis=x lineattrs=(pattern=shortdash) transparency=0.5
         noclip;
  text y=study x=xlbl text=lbl  / position=center contributeoffsets=none;
  xaxis type=log  max=100 minor display=(nolabel) valueattrs=(size=7);
  yaxis display=none  fitpolicy=none reverse valueshalign=left
        colorbands=even colorbandsattrs=Graphdatadefault(transparency=0.8)
        valueattrs=(size=7);
run;
```

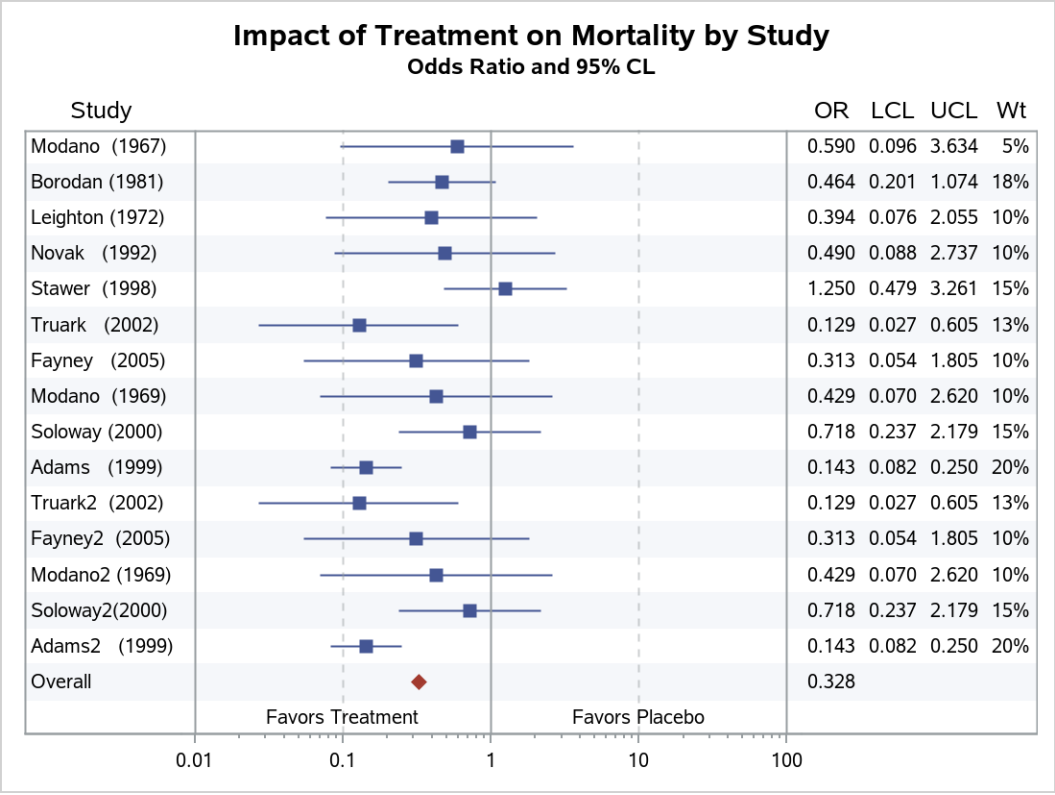

Figure 11. Forest plot

You will probably want all of the axis tables to be inside of the plot area to take advantage of color banding support. The COLORBAND option creates alternating banding along the discrete axis to make it easier to see all the plot and axis table data for each study. There are a total of five axis tables in Figure 11; but because the four tables on the right all have the same attributes, they can be combined into one YAXISTABLE statement.

The odds-ratio plot was created primarily using a grouped scatter plot, with the group used to create a different marker for the overall summary. The confidence limits were created by using the XERRORUPPER and XERRORLOWER options on the scatter plot, along with the NOERRORCAPS option to drop the whiskers on the end of the error bars. Another alternative to using the error bar options would be to overlay a high-low plot instead. There are a couple situations where this might be useful:

- You want to label the confidence limit values on the end of the bars, instead of in an axis table.
- You have a need to put an arrow on the end of the error bar if the limit extends beyond the range of the axis. Limiting the axis range can be a useful technique to see small changes when comparing the scatter points. There will be more about using arrows on high-low plots when we discuss swimmer plots later.

The two labels along the X axis were generated using the TEXT plot. The study value is a non-breaking space, so it creates blank value on the Y axis and plots the two strings along the X axis. The CONTRIBUTEOFFSETS=NONE option is used to prevent the axes from using this plot when considering offset spacing at the end of the axes. An alternative way this might be done involves using annotation for the text and using the OFFSETMAX option on the YAXIS to reserve space at the bottom of the data area for the text.

Another form of forest plot involves studies that are sub-grouped (see Figure 12). The strategy for constructing this type of forest plot is like the previous example, but there are a few additional considerations.

First, the first YAXISTABLE statement contains three options that are used to create the sub-grouping appearance in the axis table: TEXTGROUP, TEXTGROUPID, and INDENTWEIGHT.

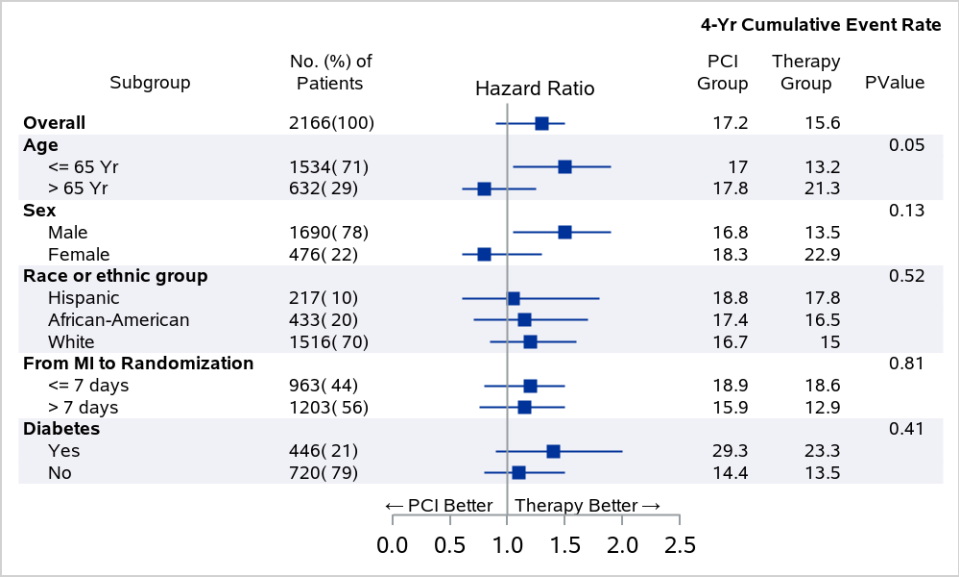

Figure 12. A sub-grouped forest plot

```
title j=r h=7pt '4-Yr Cumulative Event Rate';
ods graphics / reset width=5in height=3in imagename='Subgroup_Forest_SG_94';
proc sgplot data=forest_subgroup_2 nowall noborder nocycleattrs
dattrmap=attrmap noautolegend;
  format text $txt.;
  styleattrs axisextent=data;
  refline ref / lineattrs=(color=cxf0f0f7) DiscreteThickness=1;
  highlow y=obsid low=low high=high;
  scatter y=obsid x=mean / markerattrs=(symbol=squarefilled);
  scatter y=obsid x=mean / markerattrs=(size=0) x2axis;
  refline 1 / axis=x;
```

```
  text x=xl y=obsid text=text / position=bottom contributeoffsets=none strip;
  yaxistable subgroup  / location=inside position=left textgroup=id
             labelattrs=(size=7) textgroupid=text indentweight=indentWt;
  yaxistable countpct / location=inside position=left labelattrs=(size=7)
             valueattrs=(size=7);
  yaxistable PCIGroup group pvalue / location=inside position=right
             pad=(right=15px) labelattrs=(size=7) valueattrs=(size=7);
  yaxis  reverse display=none colorbands=odd type=discrete
         colorbandsattrs=(transparency=1);
  xaxis  display=(nolabel) values=(0.0 0.5 1.0 1.5 2.0 2.5);
  x2axis label='Hazard Ratio' display=(noline noticks novalues)
         labelattrs=(size=8);
run;
```

The TEXTGROUP option takes a grouping variable that is used to identify statement groups for attribute treatment. In the data snippet below, the ID column contains the group values for each study:

PCIGroup	Group	Id	Subgroup
17.2	15.6	1	Overall
.	.	1	Age
17.0	13.2	2	<= 65 Yr
17.8	21.3	2	> 65 Yr

The TEXTGROUPID option gives you the ability to associate a discrete attributes map to the group variable. For Figure 11, the following attributes map was used to bold the study and make is slight bigger than the sub-group:

```
data attrmap;
  length textweight $10;
  id='text'; value='1'; textcolor='Black'; textsize=7; textweight='bold';
  output;
  id='text'; value='2'; textcolor='Black'; textsize=5; textweight='normal';
  output;
run;
```

The INDENTWEIGHT option takes a column of numbers that are used as multipliers for the indention size, which defaults to 1/8 inch. This size can be changed using the INDENT option. A zero in the weight column prevent any indention from being applied.

Another important difference in this example is that using the COLORBAND option will not produce the expected coloring result. Color bands alternate colors for each discrete value on the axis. However, you will probably want the color band to include the study and all its sub-groups, so the number of items in each band can vary. To create the desired banding effect, this example uses a data driven REFLINE statement to turn on reference lines at the desired discrete values. In addition, setting DISCRETETHICKNESS=1 makes adjacent reference lines wide enough so that they touch, creating a band effect.

One other subtle technique in this example is the placing of the “Hazard Ratio” header above the plot. This header is the X2 axis label. To make this label appear, a scatter plot with zero-sized markers was added and assigned to the X2 axis, Then, the X2AXIS statement was used to display only the axis label.

PATIENT LONGITUDINAL GRAPH

The graph in Figure 13 shows a patient’s tumor cell count over time. Most of the features in the graph have already been discussed in this paper, but there is one plot type shown here that has not been discussed – the block plot:

```
Title1 h=16pt "Patient Longitudinal Graph";
Title2 h=8pt "PR=Partial Response    SD=Stable Disease by Imaging
PD=Progressive Disease by Imaging";
footnote j=l "Series and Block Plot are used to create this graph";
proc sgplot data=patlong;
  yaxis label="CTC Count / 7.5 mL of Blood" labelattrs=(weight=bold size=8)
        values=(0 to 10 by 1);
  y2axis label="CEA     U/mL" labelattrs=(weight=bold size=8)
         values=(0 to 20 by 2);
  xaxis display=(noline);
  block x=date block=blk / filltype=multicolor position=top
        valueattrs=(weight=bold ) outline values nolabel;
  block x=date block=therapy / filltype=alternate position=bottom
        valueattrs=(weight=normal size=8 ) outline fill values nolabel;
  refline '01MAY07'd '01JUL07'd '01NOV07'd / axis=x lineattrs=(thickness=2 );
  series x=date y=cea / lineattrs=(thickness=2 color=red) markers y2axis
         markerattrs=(size=9 color=red symbol=circlefilled) name="ser2"
         legendlabel="Carcinoembryonic antigen (CEA)";
  keylegend "ser1" "ser2" / location=outside border;
  series x=date y=ctc / lineattrs=(thickness=2 color=green) markers
         markerattrs=(size=9 color=green symbol=circlefilled) name="ser1"
         legendlabel="Circulating tumor cells (CTC)";
  refline 3 / label="CTC cutoff" lineattrs=(pattern=dash color=gray)
          labelattrs=(weight=bold) labelpos=min labelloc=inside;
run;
```

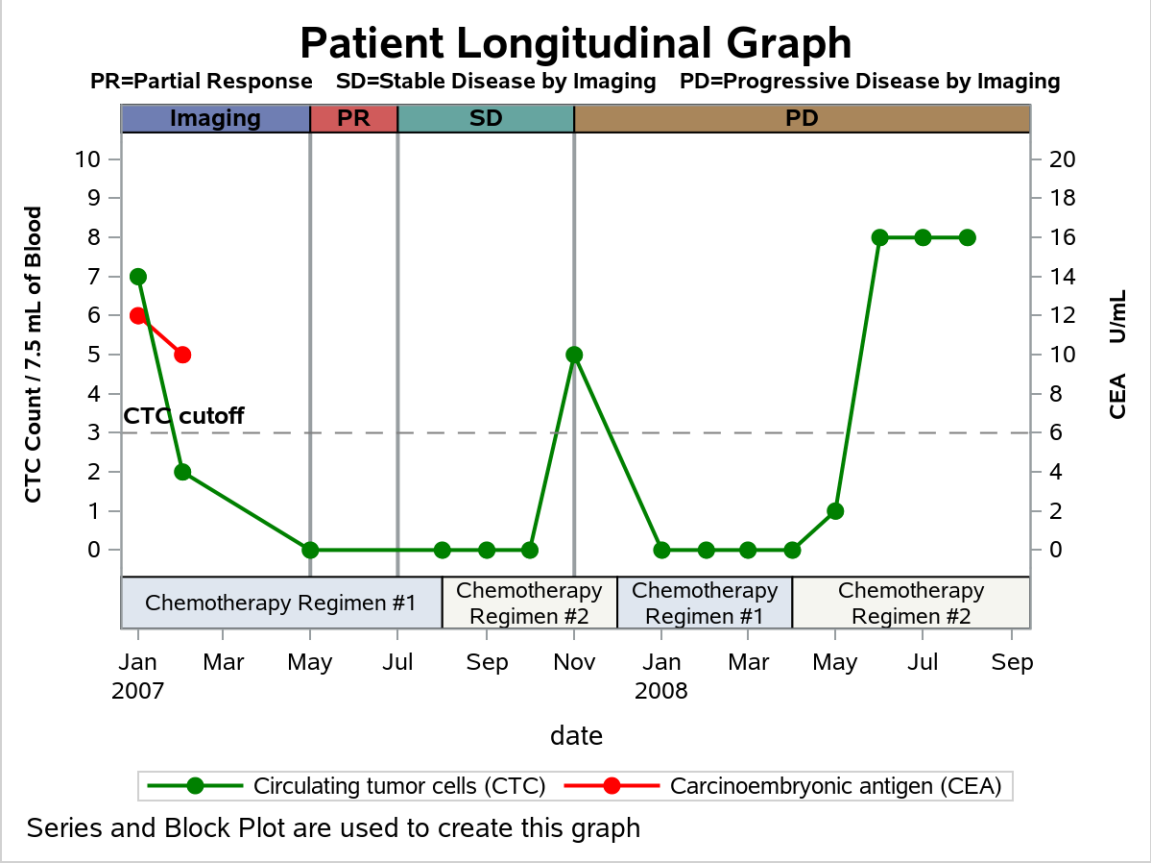

Figure 13. Patient longitudinal graph

Block plots are great for showing status changes over time. Multiple block plots can be used together to visualize multiple statuses at any point along the timeline. For example, in Figure 13, you can determine the chemotherapy regimen and the response status for any point along the series plot tracking the tumor cell count. Block plots can be stacked above or below the plot area, or a single block plot can be drawn behind other plots to show the change in status along the wall in the plot area. In addition, block plots can be colored using an alternating color scheme or a multi-color scheme by using the FILLTYPE option.

SWIMMER PLOT CONSTRUCTION

This final example brings together many of the plot types and techniques discussed in the previous examples.

The swimmer plot in Figure 14 pulls together a lot of information in one display, including the disease stage of the patient, the response type and duration, and whether the patient is a durable responder.

Analyzing this plot further, there is one plot that contains filled areas, which is created using a high-low plot with the TYPE option set to BAR. Using the ordering principles discussed earlier, this plot should be specified first to have it drawn below the other plots. On top of these bars, there is another line-type high-low plot and two scatter plots – one to show the start of the response and the other to show the end. In this case, the high-low should be drawn before the scatter plots to guarantee that there are no visual artifacts from the line on top of the markers.

Previously in the forest plot example, I mentioned the ability to add arrows on the end of high-low plots. This plot demonstrates that capability. The HIGHCAP and LOWCAP options give you the ability to add different types of caps on the end of the high-low plot, whether it be for the entire plot or controlled from the data. In this case, the feature is controlled from the data, based on whether the patient has continued response.

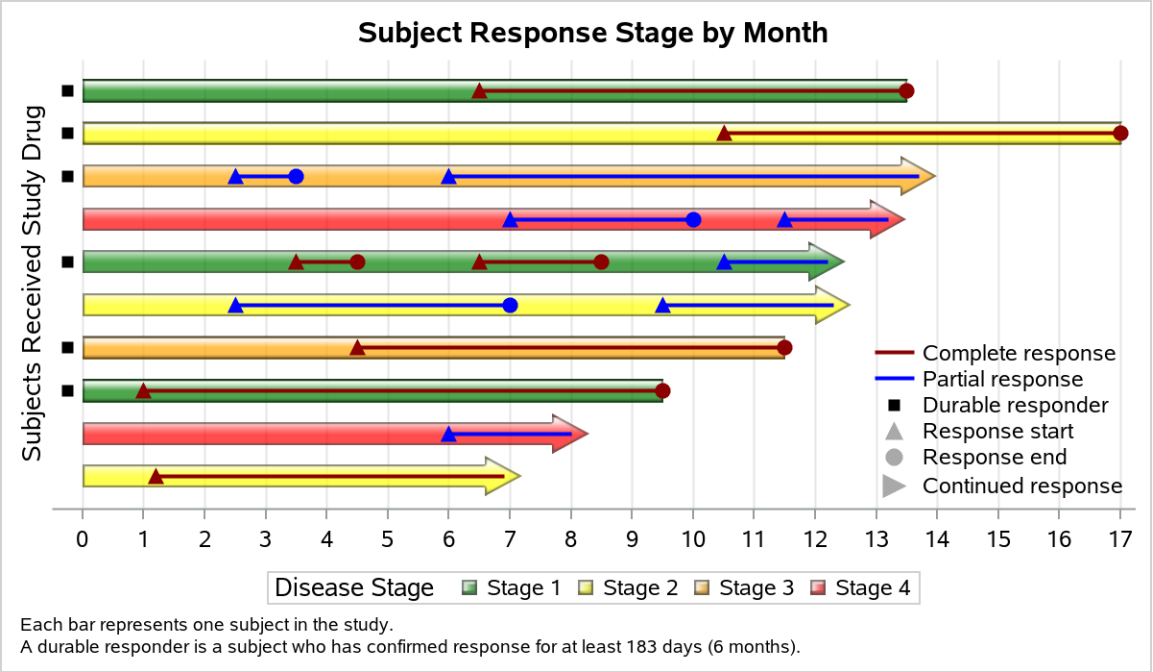

Figure 14. Swimmer plot

```
title 'Subject Response Stage by Month';
footnote  J=l h=0.8 'Each bar represents one subject in the study.';
footnote2 J=l h=0.8 'A durable responder is a subject who has confirmed
response for at least 183 days (6 months).';
proc sgplot data= swimmer dattrmap=attrmap nocycleattrs noborder;
  legenditem type=marker name='ResStart' / markerattrs=(symbol=trianglefilled
```

```
            color=darkgray size=9) label='Response start';
  legenditem type=marker name='ResEnd' / markerattrs=(symbol=circlefilled
            color=darkgray size=9) label='Response end';
  legenditem type=marker name='RightArrow' /
            markerattrs=(symbol=trianglerightfilled
            color=darkgray size=12) label='Continued response';
  highlow y=item low=low high=high / highcap=highcap type=bar group=stage
          fill nooutline dataskin=gloss lineattrs=(color=black) name='stage'
          barwidth=1 nomissinggroup transparency=0.3 attrid=stage;
  highlow y=item low=startline high=endline / group=status attrid=status
          lineattrs=(thickness=2 pattern=solid) name='status' nomissinggroup;
  scatter y=item x=durable / markerattrs=(symbol=squarefilled size=6
          color=black) name='Durable' legendlabel='Durable responder';
  scatter y=item x=start / markerattrs=(symbol=trianglefilled size=8)
          group=status attrid=status;
  scatter y=item x=end / markerattrs=(symbol=circlefilled size=8)
          group=status attrid=status;
  xaxis   display=(nolabel) values=(0 to 20 by 1) valueshint grid;
  yaxis   reverse display=(noticks novalues noline)
          label='Subjects Received Study Drug' min=1;
  keylegend 'stage' / title='Disease Stage';
  keylegend 'status' 'Durable' 'ResStart' 'ResEnd'  'RightArrow' /
            noborder location=inside position=bottomright across=1
            linelength=20;
run;
```

The technique for marking patients as durable responders takes advantage of axis tick marking behaviors in ODS Graphics. The axis has this concept of a “threshold”, where an enclosing tick mark is generated only if a data value falls within the threshold. The default threshold value is 0.30 (30%) and is changeable on both ends of the axis using the THRESHOLDMIN and THRESHOLDMAX options. In this plot, the data range between each tick mark is 1.0. The X value for all the durable scatter points is -0.25, which does not fall within the last 30% of the inter-tick data range. Therefore, no tick mark was produced.

The inside legend for this plot contains combinations of plot information and legend items. The “Response start” and “Response end” chicklets are defined as a legend item using a neutral-color symbol because the high-low legend entries show the colors associated with each response type. The legend just needs to show what the symbol shape means in the plot. Also, the high-low arrowhead is not a retrievable item by the legend, so a TRIANGLERIGHTFILLED marker was created in a legend item to represent “Continued Response” in the bar-type high-low plot.

Finally, this example uses two attribute maps – one for the disease stage and one for the response type. Both attribute maps are contained in one data set:

```
data attrmap;
length ID $ 9 linecolor markercolor fillcolor $ 10;
input id $ value $10-30 linecolor $ markercolor $ fillcolor $;
show='ATTRMAP';
datalines;
status   Complete response    darkred   darkred  gray
status   Partial response     blue      blue     gray
stage    Stage 1              darkred   darkred  green
stage    Stage 2              black     black    yellow
stage    Stage 3              black     black    orange
stage    Stage 4              black     black    red
;
run;
```

Attribute maps promote consistency in your output by assigning visual attributes to a data value (discrete attributes map) or a data range (range attributes map). These attributes are correctly applied to the plot

regardless of data order or whether all the expected data is present. Each map in this data set is referred to by its ID from a plot that is using the map. In this example, the ATTRID option is used to make this reference, but other plot options ending in "ATTRID" can also be used to refer to either a discrete attributes map or a range attributes map, depending on what is needed by the display.

The SHOW column containing the value ATTRMAP tells the plot to share all the attribute map entries to the legend, even if there is no data for that entry to be display. This gives you the ability to create consistent legend content across multiple plots, regardless of the incoming data.

CONCLUSION

As you work through creating your own displays, remember the steps discussed in this paper:

1. Break down the display into simple plot types.
2. Order those plot types so that plots containing filled areas are drawn first, unless you are using transparency on those areas for a special effect.
3. Add additional supporting statements as needed. These statements are not order-dependent, except for the PANELBY statement in the SGPANEL procedure, which must be the first statement.

Remember to think "out-of-the-box". Many plots, such as band plots, text plots, and high-low plots can be used creatively beyond their original purpose to help create effective displays for your information.

Finally, I recommend you explore the use of attribute maps in your output. Using attribute maps can help promote consistency in your output by giving you a central place to assign visual attributes to your data.

REFERENCES

Sarkar, Depriya. 2014. "Plotting Against Cancer: Creating Oncology Plots Using SAS." *Proceedings of the SAS Global 2014 Conference*, Pune, India: SAS Institute Inc. Available at https://support.sas.com/resources/papers/proceedings14/SAS130-2014.pdf.

Matange, Sanjay. "Swimmer plot." June 22, 2014. Available at https://blogs.sas.com/content/graphicallyspeaking/2014/06/22/swimmer-plot/.

Matange, Sanjay. "Clinical Graphs." November 15, 2014. Available at https://blogs.sas.com/content/graphicallyspeaking/2014/11/15/clinical-graphs/.

Hebbar, Prashant. 2015. "Lost in the Forest Plot? Follow the GTL AXISTABLE Road!" *Proceedings of the SAS Global 2015 Conference*, Cary, NC: SAS Institute Inc. Available at https://support.sas.com/resources/papers/proceedings15/SAS1748-2015.pdf.

Matange, Sanjay. "Survival plot with a twist using SGPLOT procedure." February 19, 2018. Available at https://blogs.sas.com/content/graphicallyspeaking/2018/02/19/survival-plot-twist-using-sgplot-procedure/.

Matange, Sanjay. 2016. "Clinical Graphs Using SAS." *Proceedings of the SAS Global 2016 Conference*, Cary, NC: SAS Institute Inc. Available at https://support.sas.com/resources/papers/proceedings16/SAS4321-2016.pdf.

ACKNOWLEDGMENTS

Special thanks for Sanjay Matange, Depriya Sarkar, and Prashant Hebbar for use of their examples.

RECOMMENDED READING

- *Graphically Speaking blog*
- *Clinical Graphs Using SAS*

CONTACT INFORMATION

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

Dan Heath
SAS Institute Inc.
Dan.Heath@sas.com