PharmaSUG 2015 – Paper DV06

Have a Complicated Graph? Annotate Can be Great!

Scott Burroughs, PAREXEL International, Durham, NC

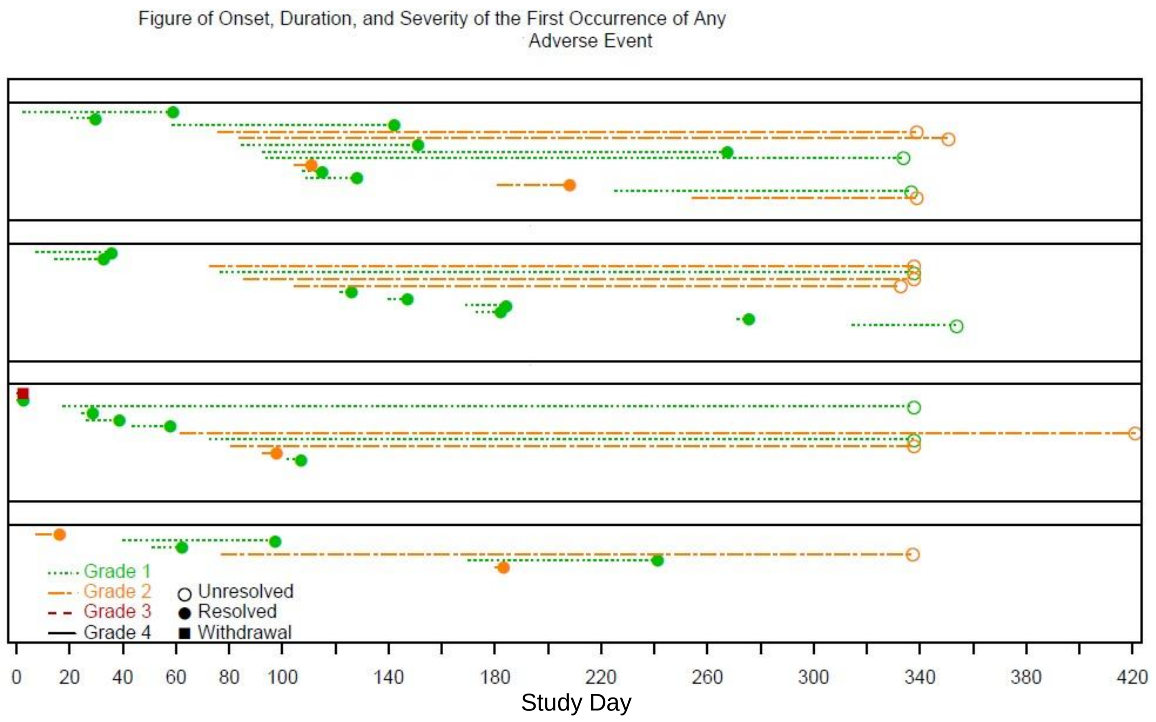
ABSTRACT

Back in the day before PROC REPORT became popular to create tables, DATA _NULL_ was the go-to vehicle to do just about anything you wanted in a table. It was versatile and highly customizable. SAS/GRAPH procedures have been adding new functions and features throughout the years, and now we have the powerful SG line of PROCs to use. However, there still are times when they can't do exactly what we want.

The ANNOTATE feature of SAS/GRAPH is still being used to append both data-driven and stand-alone objects to graphs, including data points, p-values, and other highlighting features. But could you do the entire figure using it? Certainly! This will not be my first PharmaSUG paper using ANNOTATE to do all of the data presentation.

INTRODUCTION

One of the safety graphs we have used for some of our studies is a safety signal duration plot, where you see the event occur on day X, and then the duration of the event marked by a line, followed by the result of the event (Unresolved, Resolved, W/D, etc.) at the end of the line denoted by various symbols. Also, each line type and color during the duration of the signal varied by the severity of the signal. Treatments were separated along the Y-axis, with each subject within the respective treatment groups. Multiple events per subject could be plotted along the X-axis.



We used a current standard graphing macro of course to create these, with some additional data included to create a custom legend. I first tried to use this program to get what I needed, but found all the details needed couldn't be done. Then I tried to see if it could be done using PROC GPLOT with annotation (I even tried PROC PLOT!). None of these methods could do everything I wanted (keep in mind we were still on

SAS® 9.1.3, so I didn't have the SG procedures available at the time to try). So I figured I'd have to go with the old standby, full use of annotation, as I had done a decade ago (and presented at a SUG or two).

METHODS

From the figure above, some of the differences that the new graph would have follows:

1. We didn't need the duration of one or two events per person, but every visit the subjects had from baseline to finish (or last data available)

2. Each visit needed to be present, which meant that values within normal range had its own color

3. Each phase of the study (up to 4) needed to be delineated somehow (by different line types)

When I came to the conclusion that I seemingly had to use 100% annotation to complete the graph, I knew it could be done, since I had already done it before and a few of our standard safety macros used it, as well. Although I had used it before, the X-axis and Y-axis dimensions in the first graph were essentially fixed, so the data-driven variability wasn't quite as crucial as this time. However, the variability here included: for each interim data cut, time will have elapsed, so the X-axis will need to be 'scrunched in' for the fixed width of the page. Also, additional subjects may have experienced the event, so the number of subjects along the Y-axis may be increasing. SAS PROCs usually do all this for you, but for this it would need to be programmed in to handle new/additional data each time.

As I thought of all the parts of this graph, I discovered many places that could vary/be dependent on the number of days, treatments, subjects, severity levels of the event, and time periods in the study. In order to program these variables correctly, I was going to have to do the most amount of algebra/formula equations I'd done since college.

With the fixed width and height of the graph output area, like with other times I've used annotate, I need to be using the Annotate variables XSYS and YSYS='3', the percentage of graphics output area (absolute systems). However, with study day on the x-axis and number of subjects on the y-axis increasing for each data cut, the percentage for each day/subject needs to shrink each time, so a formulae for each was needed to get a standard 'day unit' and 'subject unit', based on the maximum number among all subjects/treatments.

Building the Graph

Where do we start a graph from? The origin, of course. Everything will be built from the origin, which will need space below for tick marks, tick labels, an x-axis label, footnote(s), and the jobid, and to the left for the treatment arm labels (which were moved to the left side from the original graph to save vertical space).

The far right of the graph can't be set at 100% (as I found out through trial and error), as any x-axis label that extends beyond the right edge will get cut off. So I set it to 98%. The upper part of the graph needs to allow for at least 4 title lines. These outer edges will be used to calculate midpoints and/or quarter points along each axis for labels (x-axis) and treatment lines/labels (4 treatment arms), not to mention 'origins' for each treatment vertically.

<u>Basics of Annotate</u>

The main variables needed for most ANNOTATE data sets include: XSYS, YSYS, X, Y, FUNCTION, POSITION, TEXT, STYLE, LINE, COLOR, and SIZE.

XSYS and YSYS are the coordinate systems as mentioned above.

X and Y are of course the coordinates of the graphing area.

FUNCTION dictates whether you want to 'move', 'draw' (a line), or 'label' at the certain coordinate.

POSITION dictates where (on a ‘telephone keypad’ direction relative to the coordinate you want to ‘draw’ or ‘label’ something. ‘5’=centered, ‘1’=upper left, etc.)

TEXT what you print when you use ‘label’ function.

STYLE is the text font.

LINE is the type of line when you use the ‘draw’ function. ‘1’=solid and all others are patterns of dotted lines.

COLOR is the color of whatever you annotate as ‘text’ or ‘label’.

SIZE is the thickness of the LINE or how big your ‘label’ text is.

All of the frames, treatment section dividers, and tick marks of the graph are simply ‘move’ and ‘draw’ functions.

All of the tick and treatment labels are simple ‘move’ and ‘label’ functions. The rest (visits + duration lines and all of the legend) are ‘move’, ‘draw’, and ‘label’ functions.

The Data Lines/Visits

I decided that the way I was going to output the data to show all of the visits and time lines was to output each as two dates (study days), with a line between them, and a small circle at the 2nd visit. The first visit would just be the same study day for both the first and second visits (thus no line drawn). The color of the line is determined by the grade of the event. If a change in line type (phase of study) occurred between visits, then two sets of start and stop days were output (each with different line types), with no visit circle printed at the change day.

Other data points to be added to the graph are the day of withdrawal (denoted by a red circle) and the day of discontinuation of treatment (denoted by a black vertical line).

Legend

Luckily one of the treatment arms had few enough events (continually) to be able to fit the legend within its quarter (section) of the graph, without sacrificing space between subjects. In order to put the W/D day and treatment D/C symbols in the legend and keep the same data structure, I made the color of the lines white and second dot white, so only the symbol was visible.

The grade of the event for each time between visits was denoted the variable ‘group’ in my data sets, and group determined many factors of the graph, such as line color (in the treatment section and separately in the legend), line type, x and y coordinates in the legend, and labels in the legend. Because of this, I created a format for each type, so they can be controlled in one place.

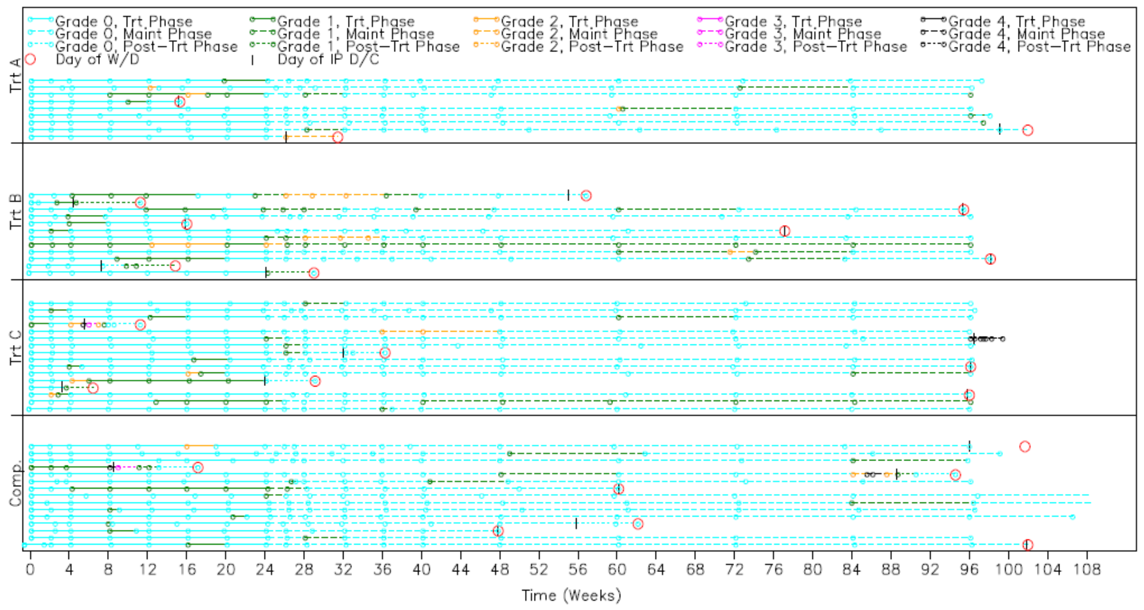
```
proc format;
  value color
    1,6,10,18='cyan'
    2,7,11,19='green'
    3,8,12,20='orange'
    4,13,14,21='magenta'
    5,15,16,17,22='black'
    9='red'
    ;
  value colorw
    1,6,10,18='cyan'
    2,7,11,19='green'
    3,8,12,20='orange'
    4,13,14,21='magenta'
    5,16,17,22='black'
    9,15='white'
    ;
  value lyne
```

```
    1,2,3,4,5='1'
    6,7,8,9,14,15,16='20'
    10,11,12,13,17='35'
    18,19,20,21,22='35'
    ;
  value xinc
    1,6,10,9,18='0'
    2,7,11,15,19="&xinc1"
    3,8,12,20="&xinc2"
    4,14,13,21="&xinc3"
    5,16,17,22="&xinc4"
    ;
  value yinc
    1,2,3,4,5='0'
    6,7,8,14,16='1.65'
    10,11,12,13,17='3.3'
    18,19,20,21,22='4.95'
    9,15='6.6'
    ;
  value trtlbl
    1='Grade 0, Trt Phase'
    2='Grade 1, Trt Phase'
    3='Grade 2, Trt Phase'
    4='Grade 3, Trt Phase'
    5='Grade 4, Trt Phase'
    6='Grade 0, Maint Phase'
    7='Grade 1, Maint Phase'
    8='Grade 2, Maint Phase'
   14='Grade 3, Maint Phase'
   16='Grade 4, Maint Phase'
    9='Day of W/D'
   15='Day of IP D/C'
    10='Grade 0, Post-Trt Phase'
    11='Grade 1, Post-Trt Phase'
    12='Grade 2, Post-Trt Phase'
    13='Grade 3, Post-Trt Phase'
    17='Grade 4, Post-Trt Phase'
    18='Grade 0, Open-Label Phase'
    19='Grade 1, Open-Label Phase'
    20='Grade 2, Open-Label Phase'
    21='Grade 3, Open-Label Phase'
    22='Grade 4, Open-Label Phase'
    ;
run;
```

The finished product looks like this:

Protocol: XYZPDQ **Page 1 of 1**
Population: Safety

Figure 1
Individual Subject Profiles of Severity and Duration in Subjects With Elevations from Baseline



Note: Figure shows multiple events per subject, if applicable. Small circles are lab visits.
~/lb_f_pharmasug.sas

Data Structure

I made the data sets horizontal to do all of this, where study day, event grade, and phase for each visit were variables, in addition to W/D date, and treatment end date.

Several different parts of graph were made using algebraic formulae based on the data. Each new time I visited the graph for a new report, I automated more and more to be data driven. My goal is to get the program as close to 100% data-driven as possible.

CONCLUSION

I don't really know if this graph could be done using the SAS SG procedures (and I haven't researched it), as I started before we had production access to SAS® 9.3 (that would be a way to really become familiar with a lot of what it has to offer). In the mean time, good ole Annotate is still there to help us when a graph can get complicated.

REFERENCES

SAS® is a registered trademark of SAS Institute, Inc. in the USA and other countries.

CONTACT INFORMATION

BIOGRAPHY

Scott was a statistician for GlaxoSmithKline for almost 12 years until switching to a full-time programmer role in 2006. His entire department was moved to PAREXEL International in March, 2105. He has worked in Research Triangle Park, NC since 1994. He has programmed in SAS extensively since 1992 while at a previous pharmaceutical company. He has a B.S. and an M.S. in Statistics from Virginia Tech.

CONTACT

Scott Burroughs
PAREXEL International
1818 Ellis Rd.
Durham, NC 27703
scott.burroughs@parexel.com