The Rest of the Way There…
Creating Multiple Graphs on a Single Page
Mary Buckwalter, Shire Pharmaceuticals, Wayne, PA
Terek Peterson, Octagon Research Solutions, Wayne, PA

ABSTRACT
Have you ever felt like you have been left in the woods when trying to find solutions to your SAS® coding problems? We all know using graphs to convey information is powerful, but producing the right one can be a bear. This paper will lead you through the trees to uncover dynamic solutions to creating multiple graphs on a page with a single title and common legend. This paper will also uncover interesting code gnomes that present graphs in high quality. This paper is for intermediate SAS users, even if the authors have the knowledge of a troll when it comes to SAS/Graph.

INTRODUCTION
All SAS programmers quickly realize that it is impossible to learn everything about SAS code syntax to get the results we want. A task such as producing multiple graphs per page (see Figure 1) would force most of us to look to our favorite resources for help. Those who have been in the business for a while may turn to the volumes of books they carry from job to job with their favorite sections flagged. Others may turn to SAS Help or SAS OnlineDocs to find that line of code that may need just a little tweaking to meet the customers needs. Savvy programmers may read conference proceedings, use SAS-L, call the SAS help desk, or simply Google to get that perfect solution that SAS does not necessarily provide in the electronic documentation or SAS books. Our goal is not to simply provide you the solution to multiple graphs per page, but to get you thinking about how to solve your next SAS problem using the research techniques we used. Although these techniques may not be exactly what you need, it should help you think about how to get the rest of the way there.

Figure 1
Peanut Butter Allergy and Concentration versus Time
Treatment: SPD-999  Analyte: Fiaa Stuff

And Now, Presenting... NESUG 2008
THE GOAL
Graphs are a powerful way of communicating information. The ability to produce a single graph per page using SAS/Graph is a relatively simple task for the average SAS user. Alternatively, the ability to produce multiple graphs on a single page using SAS/Graph may be a little trickier for even seasoned professionals. This is because learning to use GREPLAY and the associate coordinate system is not necessarily intuitive. In addition, there always seems to be technical trade-offs with issues like high resolution that create files that are unwieldy in size. As mentioned in the introduction, turning to SAS Help for answers often leaves one frustrated and confused.

The ultimate solution would be to dynamically produce multiple graphs per page, with a shared title and legend so that everyone from your novice programmers to seasoned professionals can use with relative ease. The macro needs to be flexible enough to dynamically change the number of graphs presented on a page without much rework. To solve this conundrum, Internet searches uncovered a macro (on the SAS website) that dynamically produces multiple graphs per page; however each individual graph had its own title. Another macro (also on the SAS website) had code that had a shared title and common legend that was needed. Although blending the two techniques from the two macros brought us closer to the desired result, it was not exactly what was needed. What follows in this paper are the steps and tricks to get us the rest of the way there.

STEP 1 – INDIVIDUAL PLOTS
The first step is to generate individual plots. The code to produce a single plot for this project is below. For this example, a plot is created for each subject in the study as can be seen by the use of the TITLE1 statement. To remove the legend from each plot the NOLEGEND option in the PLOT statement was used (see Figure 2).

```
options nobyline;
proc gplot data=p_butter;
  by treatment analyte center subno;
  plot allergy*time=1 dummy*time/haxis=axis4 vaxis=axis3 overlay nolegend;
  plot2 conc*time=3/ vaxis=axis5;
  title1 "Center: #byval(center) Subject: #byval(subno)";
run;
```

Figure 2

STEP 2 - GENERATE COMMON LEGEND
After the individual plots are generated, a second plot needs to be created which just generates the shared legend (see Figure 3 on the next page). The key parts of this step are to define the legend, remove the frame and axis from the plot.

```
proc gplot data=p_butter;
  plot allergy*time=1 dummy*time/haxis=axis100 vaxis=axis100 overlay noaxis legend=legend1 noframe;
  plot2 conc*time=3/ vaxis=axis100 noaxes noframe;
  title; footnote;
```
STEP 3 - SHARED TITLE AND FOOTNOTES
The GSLIDE procedure generates the shared titles and footnotes. The syntax below was quite easy and produced what is needed. Overlay of the individual graphs in the white space between the titles and footnotes should now work. As can be seen in Figure 4 below there was plenty of white space to do this. However, this did not give the expected outcome.

```
proc gslide ;
title1 "Peanut Butter Allergy and Concentration versus Time";
title2 "Treatment: SPD-999  Analyte: Fun Stuff";
footnote1 j=l " Programmer: Mary Buckwalter" j=c u=3 "CONFIDENTIAL" j=r "May 16, 2007  ";
run;
```

STEP 4 – PUTTING IT ALL TOGETHER
Unfortunately, using GREPLAY within the macro to combine Steps 1, 2, and 3 above lead to significant overlap in the pieces. Not only did the macro fill the entire page, the title and legend were not consistently appearing. This is where the macros on the SAS website basically left off. They correctly assumed that most programmers would want to use the entire area of the page to calculate the coordinates dynamically for the individual graphs. Now how can we trick the macro into using less of the page for the individual graphs and keep some for the shared title, shared footnote, and shared legend?

DYNAMICALLY ASSIGNING COORDINATES
Understanding the SAS coordinate system was crucial in put all of these pieces together. Additionally, it was discovered that the title and legend need to be fixed to replay 999:gslide.

The print area on a page when using GREPLAY can be understood as shown in Figure 5 on the next page. The area is a combination of 4 coordinates (ULX, ULY)=Upper Left, (URX, URY)=Upper Right, (LLX, LLY)=Lower Left, and (LRX, LRY)=Lower Right.
Now to trick Greplay, a portion of the page must be used for the title and a portion for the footnote and legend. In the example below, 10% is used for the title and 10% is used for the footnote and legend, leaving 80% of the page to display all the individual graphs. These values are used in the data step code below before dynamically calculating the coordinates. Since this was part of a macro the macro parameters &DOWN and &ACROSS were replaced by 3 and 3 respectively to produce the 3 by 3 graph in Figure 6.

```plaintext
/* Reduce the amount of printable area by the area used for titles, footnotes, & legends. */
top=90;
bottom=10;
ymult=(top-bottom)/&down; /* =3 */
xmult=100/&across;        /* =3 */
/* Dynamically calculate the coordinates */
do x=0 to (100-xmult) by xmult;
   llx=x;
   ulx=x;
   urx=x+xmult;
   lrx=x+xmult;
   do y=10 to (100-ymult) by ymult;
      lly=y;
      uly=y+ymult;
      ury=y+ymult;
      lry=y;
      output;
   end;
end;
run;
```

### FIGURE 6

**Peanut Butter Allergy and Concentration versus Time**

- Treatment: SPD-999 Analogs: Pan Stuff

And Now, Presenting ...
FIXING THE LEGEND WITH FULL TEXT CENTERED

Despite understanding the coordinate system, the legend still appeared to be compressed. It was discovered that the legend needed to be tricked in order to make it legible. By employing multiple "JUSTIFY=C" options with text in quotes created two lines. In order to increase legibility, simply adding additional spaces before and after the text and using the ORIGIN option centered the legend and took care of any compression issues. The sample syntax below combines all of these tricks:

```plaintext
legend1 label=(j=c ' ' j=c ' ') shape=symbol(5,1)
value=(tick=1 justify=c h=1 ' Peanut Butter - ' justify=c h=1 ' Allergy '
        tick=2 justify=c h=1 ' Concentration ' justify=c h=1 ' (ng/mL) ')
origin=(30pct,4pct);
```

PROBLEMS WITH HIGH RESOLUTION FILE SIZES

There are trade-offs between file size and resolution when creating numerous graphs. A balance needed to be determined so that the size of the file could be transported and printed easily with high enough resolution so details could be viewed. A massive file with high resolution is difficult to transport and print. A small file size with low resolution is difficult to view all graphic details.

Another challenge was our friends at SAS have chosen a default image type and output destination that is not optimal for this exercise. In order to get precisely the resolution and file size balance actively managing the GOPTIONS, ODS destination, and X/Y Pixels to created high resolution output that has a manageable size. To do this use ODS RTF and PNG as your DEVICE and TARGET to control the resolution. Then pick a resolution, e.g. 200 DPI; this was used for these graphs. Calculate the XPIXELS and YPIXELS these must match the size used in the XMAX and YMAX goptions statements.

```plaintext
ods rtf file="x:\peanutbutter.doc";
goptions noprompt ftext="Arial" device=png target=png
    xmax=7in ymax=9.0in xpixels=1400 ypixels=1800;
```

UNDERSTANDING XMAX, YMAX, XPIXELS, YPIXELS

To understand XPIXELS and YPIXELS is quite easy but was something that does not seem to be easy to find in the documentation. As can be seen in example below, choose the DPI (dots per inch) and the XMAX (page width less the margins) and multiple them together to get the XPIXELS. Likewise, multiple the DPI times the YMAX (page height less the margins) to get the YPIXELS.

<table>
<thead>
<tr>
<th>XMAX</th>
<th>X</th>
<th>DPI = XPIXELS</th>
</tr>
</thead>
<tbody>
<tr>
<td>7in</td>
<td>X</td>
<td>200dpi = 1400</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>YMAX</th>
<th>X</th>
<th>DPI = YPIXELS</th>
</tr>
</thead>
<tbody>
<tr>
<td>9in</td>
<td>X</td>
<td>200dpi = 1800</td>
</tr>
</tbody>
</table>

CONCLUSION

Although this task seemed overwhelming at the start, we were able to develop a simple macro that dynamically produces multiple graphs per page, with a shared title and legend that everyone from novice programmers to the seasoned professionals can use it with relative ease. The steps and tricks outlined in this paper showed how we combined two existing, publicly available, SAS macros to make it all work.
REFERENCES
http://ftp.sas.com/techsup/download/sample/graph/greplay-examples-list.html#shlegend

Delaney, Kevin P. (2003) Multiple Graphs on One page, the easy way (PDF) and the hard way (RTF). Proceedings of the Twenty-Eighth Annual SAS® Users Group International Conference, March 2003

ACKNOWLEDGMENTS
The authors would like to thank Shire Pharmaceuticals and Octagon Research Solutions for encouraging and sponsoring innovation through the use of SAS software.

CONTACT INFORMATION
Mary Buckwalter, MS
Shire Pharmaceuticals
725 Chesterbrook Boulevard
Wayne, PA 19087 USA
(484) 595-8984
mbuckwalter@shire.com

Terek J. Peterson, MBA
Octagon Research Solutions
585 East Swedesford Road, Suite 200
Wayne, PA 19087 USA
(610) 535-6500 x5658
tpeterson@octagonresearch.com

COPYRIGHT INFORMATION
SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.