Paper 135-27

Seeing Graphical Representations Clearly to Avoid Eye Strain

James D. Gilbert, Venturi Technology Partners, Kalamazoo, MI

ABSTRACT

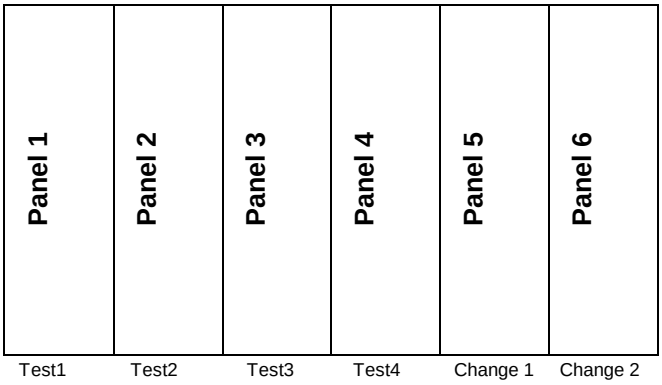
The display of multiple-group graphs, side-by-side, for fast and easy deciphering, has in the past provided a difficult task. Traditionally, there were two methods of analyzing large amounts of data both with limitations. The first, group processing, end-users were forced to look across many pages of graphs, searching for and comparing like data points. The second, custom templates and Proc Greplay, demands more intensive programming resources and the end-user still has to compare graphs on different axis.

Using data manipulation techniques enables the data to be presented as a multiple-group graph displayed in a side-by-side format.

WHERE DID THE DATA COME FROM

The data is randomly generated to emulate four unique tests, each containing two p-values and two changes over time that contain one p-value, for a total of three unique p-values. These measurements would typically be found on a number of different tables with each of the different p-values having different names.

VISUALIZE THE OUTPUT



The above graphic illustrates six distinct groups, the four tests and two changes over time. The first four panels will contain two p-values each which will be referred to as p-value1 and p-value2, while the last two panels will contain one p-value each, p-value3.

MASSAGING THE DATA

To make things as easy as possible, think of each line as a separate data set that is going to be standardized to look like each other and then be set together in a vertical format.

The process adds a distinct line number, an x-axis value, minimum and maximum values for the x-axis, and adds a blank observation at the end of each dataset.

The code below is for the first two panels and the last two panels. The macro blankobs adds a blank observation to the end of the dataset. The macro variable linemax is a count holder between panels.

```
*-------------------------------------------
 Macro blnkobs adds a blank observation to the
 end of an existing dataset.
-------------------------------------------;

%macro blnkobs (dsin=,dsout=blnkobs);
  proc sql;
    create table &dsout
    like &dsin;
    insert into &dsout
    ;
  quit;
%mend blnkobs;

*-------------------------------------------
 Get data for line 1.  Add line number, blank
 observation and xaxis count to dataset.
 Goes into Panel 1. From previous graphic.
-------------------------------------------;
data cell1a (keep=gene line snp pvalue1
             rename=(pvalue1=yaxis));
  set sugi.table1;
  line=1;
run;

%blnkobs (dsin=cell1a);

data cell1a;
  set cell1a blnkobs;
  xaxis=_N_;
run;

proc sql noprint;
  create table cell1a as
  select *, max(xaxis) as maxax, min(xaxis) as
    minax
  from cell1a;
quit;

*-------------------------------------------
 Get data for line 2.  Add line number, blank
 observation and xaxis count to dataset.
 Goes into Panel 1. From previous graphic.
-------------------------------------------;
data cell1b (keep=gene line snp pvalue2
             rename=(pvalue2=yaxis));
  set sugi.table1;
  line=2;
run;

%blnkobs (dsin=cell1b);

data cell1b;
  set cell1b blnkobs;
  xaxis=_N_;
run;

proc sql noprint;
  create table cell1b as
  select *, max(xaxis) as maxax, min(xaxis) as
    minax
  from cell1b;
quit;

*-------------------------------------------
 Get data for line 3.  Add line number, blank
 observation and xaxis count to dataset.
 Goes into Panel 2. From previous graphic.
-------------------------------------------;
proc sql noprint;
```

```
  select max(xaxis) into :linemax
  from cell1a;
quit;

data cell2a (keep=gene line snp pvalue1
             rename=(pvalue1=yaxis));
  set sugi.table2;
  line=3;
run;

%blnkobs (dsin=cell2a);

data cell2a;
  set cell2a blnkobs;
  xaxis=_N_+&linemax;
run;

proc sql noprint;
  create table cell2a as
  select *, max(xaxis) as maxax, min(xaxis) as
    minax
  from cell2a;
quit;

*-----------------------------------------------
Get data for line 4.  Add line number, blank
observation and xaxis count to dataset.
Goes into Panel 2. From previous graphic.
-----------------------------------------------;
data cell2b (keep=gene line snp pvalue2
             rename=(pvalue2=yaxis));
  set sugi.table2;
  line=4;
run;

%blnkobs (dsin=cell2b);

data cell2b;
  set cell2b blnkobs;
  xaxis=_N_+&linemax;
run;

proc sql noprint;
  create table cell2b as
  select *, max(xaxis) as maxax, min(xaxis) as
    minax
  from cell2b;
quit;


              .
              .
              .

*-----------------------------------------------
 Get data for line 9.  Add line number, blank
 observation and xaxis count to dataset.
 Goes into Panel 5. From previous graphic.
-----------------------------------------------;
proc sql noprint;
  select max(xaxis) into :linemax
  from cell4a;
quit;

data cell5a (keep=gene line snp pvalue3
             rename=(pvalue3=yaxis));
  set sugi.chng1;
  line=9;
run;

%blnkobs (dsin=cell5a);

data cell5a;
set cell5a blnkobs;
  xaxis=_N_+&linemax;
run;

proc sql noprint;
  create table cell5a as
  select *, max(xaxis) as maxax, min(xaxis) as
    minax
  from cell5a;
quit;

*-----------------------------------------------
 Get data for line 10.  Add line number, blank
 observation and xaxis count to dataset.
 Goes into Panel 1. From previous graphic.
-----------------------------------------------;
proc sql noprint;
  select max(xaxis) into :linemax
  from cell5a;
quit;

data cell6a (keep=gene line snp pvalue3
             rename=(pvalue3=yaxis));
  set sugi.chng2;
  line=10;
run;

%blnkobs (dsin=cell6a);

data cell6a;
  set cell6a blnkobs;
  xaxis=_N_+&linemax;
run;

proc sql noprint;
  create table cell6a as
  select *, max(xaxis) as maxax, min(xaxis) as
    minax
  from cell6a;
quit;

*-----------------------------------------------
Set datasets together and transform pvalues.
-----------------------------------------------;
data sugi.graphit;
  set cell1a cell1b cell2a cell2b . . .
      cell5a cell6a;
  yaxis2=abs(log10(yaxis));
  xaxis2=xaxis;
run;
```

This code is one way of creating a standardized data set that is appropriate for creating graphs similar to what was described in the visualizing the output portion of this paper. NOTE: yaxis2 is the value to be graphed on the y-axis and xaxis2 is created for use in drawing the vertical lines later in the process. There is no relationship between yaxis2 and xaxis2.

GOPTIONS, AXIS STATEMENTS AND SYMBOLS

The device is set up to use psepsf, which is an encapsulated postscript program.

```
filename gsasfile "C:\sugi\plotxy.eps";

goptions  reset  = all
          device = psepsf
          gunit  = inches
          cback  = white
          gaccess= gsasfile
          hsize  = 8 in
          vsize  = 6 in
          ftitle = HWPSL005
          htitle = .1375 in
          ftext  = HWPSL005
          htext  = .1375 in
          ;
run ;


*-----------------------------------------------
Assign macro variables to be used in creation
of x and y axis.
```

```
-------------------------------------------;
proc sql noprint;
  select max(yaxis2)+3.5 into :maxy
  from sugi.graphit;
  select max(xaxis) into :maxx
  from sugi.graphit;
  select min(xaxis) into :minx
  from sugi.graphit;
quit;

*-------------------------------------------
 Define axis.
-------------------------------------------;
axis1
  order=(0 to &maxy by 1)
  label=(font=HWPSL005 height=.1375 in angle =
         90 "Abs(log10(P-value))")
  minor = (number=4)
;

axis2
  order=(&minx to &maxx by 1)
  label=(font=HWPSL005 height=.1375 " ")
  minor = none
;

*-------------------------------------------
 Define Symbols. NOTE:  A symbol has to be
 defined for each line.
 Square associated with p-value1.
 Dot associated with p-value2.
 Circle associated with p-value3.
-------------------------------------------;
symbol1
  value = square
  interpol = join
  line = 1
  color = black
  width = .1
  height = .1
;

symbol2
  value = dot
  interpol = join
  line = 20
  color = black
  width = .1
  height = .1
;
symbol9
  value = circle
  interpol = join
  line = 1
  color = black
  width = .1
  height = .1
;
```

X-AXIS

The x-axis, at this point, is simply a number for each of the points on the graph, which mean absolutely nothing. To make this meaningful, each of the panels needs to have a name, and the numbering scheme of 1 to n needs to go away. The quick and dirty way to make this happen is to format them to missing, making sure that your options for missing is set to ' '. Once the meaningless numbers are gone, annotate is used to create meaningful labels for each of the panels on the x-axis.

```
*-------------------------------------------
 Add text to annotate datasets for x-axis
 labels. Determine minimum and maximums for
 use in positioning labels.
-------------------------------------------;
proc sql noprint;
  create table annolbl0 as
  select unique(xaxis), minax, maxax,
         maxax-((maxax-minax)/2) as x
  from sugi.graphit;
  create table annolblx as
  select unique(x)
  from annolbl0;
  select count(unique maxax) into :maxax1
  from sugi.graphit;
quit;

data xtext;
  length text $60;
  text='Test 1 Responder';
  output;
  text='Test 2 Responder';
  output;
  text='Test 3 Responder';
  output;
  text='Test 4 Responder';
  output;
  text='Test 1 Chg. From Base';
  output;
  text='Test 2 Chg. From Base';
  output;
run;

data anolabel;
  merge annolblx xtext;
run;

data annoaxis;
  set anolabel;
  length function $8 text $60;
  retain xsys '2' ysys '5' style "&font1"
         angle 0 function 'label' size .8
         position "5" ;
  y=6;
run;
```

LEGEND

The nolegend option in proc gplot is turned off and the legend is created using annotate. If proc gplot where allowed to generate the legend, ten lines would be generated. This would be repetitive and unnecessary since three types of lines can represent all ten lines.

The following code provides annotate data that performs the task of creating the legend.

```
*-------------------------------------------
 Add Legend to bottom of Graph.
-------------------------------------------;
data anoleg;
  length text $9. color style function $8;
  retain xsys ysys '3' ;
  color='black';
  function='symbol';
  size=.8;
  x=26; y=2;text='square';       output;
  x=55; y=2;text='dot';          output;
  x=84; y=2;text='circle';       output;
  x=28; y=2;text='square';       output;
  x=57; y=2;text='dot';          output;
  x=86; y=2;text='circle';       output;
  x=30; y=2;text='square';       output;
  x=59; y=2;text='dot';          output;
  x=88; y=2;text='circle';       output;

run;

data anoleg1;
  %annomac;
  %dclanno;
  %system(3,3);
  *** draw box around legend;
  %move(10,4);
```

```
  %draw(90,4,black,1,0) ;
  %draw(90,1,black,1,0) ;
  %draw(10,1,black,1,0) ;
  %draw(10,4,black,1,0) ;
  *** draw lines ;
  %move(25,2);
  %draw(31,2,black,1,0) ;
  %move(54,2);
  %draw(60,2,black,20,0) ;
  %move(83,2);
  %draw(89,2,black,1,0) ;
  %label(11,3,'Allelle P-Value',
         black,0,0,.8,zapf,6);
  %label(40,3,'Overall P-Value',
         black,0,0,.8,zapf,6);
  %label(69,3,'P-Values Between',
         black,0,0,.8,zapf,6);
run;

data anolegu;
  set anoleg1 anoleg ;
run;
```

ADDING BELLS AND WHISTLES

In this example, the identity of significant values where annotated into the body of the graph.

```
data annoit ;
  set sugi.graphit;
  length function $8 text $60;
  retain xsys ysys '2' style "&font1" angle 90
         function 'label' size .8 position "5" ;
      if yaxis2 gt 1.3 then
        do;
          if (mod(panel,2)=0) then
            do;
              grpnum3=panel-1;
            end;
          else do;
              grpnum3=panel;
          end;
          x=xaxis;
          y=yaxis2+.5;
          text=snp;
          output;
        end;
run;

proc sort data=annoit out=anno2;
  by snp grpnum3 y;
run;

data anno3;
  set anno2;
  by snp grpnum3 y;
  if last.grpnum3 then output;
run;
```

There are three distinct areas of annotation, the x-axis, the legend, and the significant values. Each of these areas was done separately and need to be set together for use in the Proc Gplot procedure.

```
data annouse;
  set anno3 annoaxis anolegu;
run;
```

ADDING VERTCAL LINES

The vertical lines are added using a macro variable.

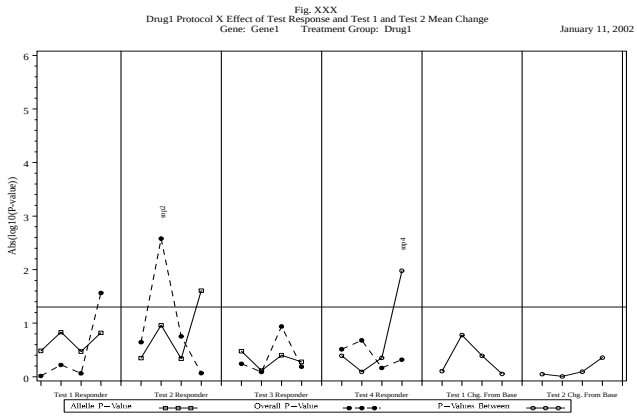
```
proc sql noprint;
  select unique(xaxis2) into :vertline separated
by ' '
  from sugi.graphit
  where yaxis eq . ;
quit;
```

GPLOT

In the gplot the reference line is added at a level of significance.

```
proc gplot data=sugi.graphit anno=annouse;
   plot yaxis2*xaxis=line/nolegend vaxis=axis1
haxis=axis2 href = &vertline vref=1.3;
run ;
quit ;
```

OUTPUT



CONCLUSION

Through the standardization of data it is possible to display multiple group graphs in a side-by-side format with relative ease.

CONTACT

James D. Gilbert
Venturi Technology Partners
5278 Lovers Lane
Kalamazoo, MI 49004
Work Phone: (616) 344-4100
Email: JimJen2727@aol.com