Paper 30-26

Using SAS/Graphs in Frames

By Phil Mason, Wood Street Consultants Ltd., England

Overview

This paper describes a technique that allows you to use some SAS/Graphs, such as Gplots and Gcharts more effectively in Frame applications using the SAS/AF product. The technique particularly describe how to detect the part of a graph that is clicked on, so that drill down and other data related functions can be implemented.

What additional features can SAS/Graph give you

Although AF graphics widgets can display data in a graphical form, and allow information to be returned relating to what point or bar was clicked on, there are several additional things that my method provide. My technique has these benefits:

1. Two (or more) y-axes can be used on graphs
2. Can detect when you click near a point, but not quite on it
3. Can detect when there are several points on top of each other
4. Far more control over layout of axes, labelling, font size, etc.
5. Far more control over symbols for points, lines, etc.
6. Ability to add custom graphics, text using DSGI or Annotate
7. Ability to produce custom graphics of virtually any kind

Why doesn't everyone use SAS/Graphs then?

Graphics widgets are easier to use, and if they satisfy your requirements then simply use them. SAS/Graphs can be displayed in a SAS/Graph Output widget, however when clicking on the graph you cannot get the same degree of information returned.

Clicking on a Graphics Widget can return:

- type of graph clicked on (e.g. hbar, join)
- id or subgroup value
- name, type & value of x-axis variable
- name, type & value of y-axis variable
- text associated with point, if there is any

Clicking on a SAS/Graph output widget can return:

- information about a pre-defined hotspot
- a number indicating the segment of the graph clicked on
- text, if there is text associated with the segment clicked on
- x & y coordinates (in pixels) of where the click was

So if you use a graphics widget to do a simple line plot then you can click on a point on the graph and get the x & y values, which can then be used for drilling down. However if you do this with the output of a PROC GPLOT displayed in a SAS/Graph Output widget then all you can get is a segment number (e.g. 27) and the x/y coordinates of where you clicked.

How to use the technique

Basically the technique involves taking control of your SAS/Graph generation so that you know exactly where the axes are positioned. Then by doing some calculations you are able to convert the x/y pixel coordinates of a click into x/y percentage coordinates. Knowing that a user clicked 60% across and 10% up from the axis origin enables you to easily determine the data plotted at that point, or near that point.

Here are the basic steps:

1. Create a Proc Gplot, specifying where the axes begin and how long they are.
    a. Use an AXIS statement for the x-axis
        i. Use ORIGIN= to specify the origin of the x and y axes in percent
        ii. Use ORDER to specify the range of values on the x-axis
        iii. Use LENGTH= to specify the x-axis length in percent
        iv. Specify OFFSET=(0,0) so that data values are not offset, but plotted directly on the x and y axes – this makes calculations easier
    b. Use an AXIS statement for the y-axis
        i. Use ORIGIN= to specify the origin of the x and y axes in percent, making sure it is the same as the x-axis specification
        ii. Use ORDER to specify the range of values on the y-axis
        iii. Use LENGTH= to specify the y-axis length in percent
        iv. Specify OFFSET=(0,0) so that data values are not offset, but plotted directly on the x and y axes – this makes calculations easier
    c. Specify GOPTIONS NODISPLAY, so that the graph is not displayed but just written to a catalog member
    d. Generate the graph, writing the output to a catalog member, and using the x & y axis specifications
    e. Specify GOPTIONS DISPLAY, so that the graph can be displayed in the frame
2. Define a SAS/Graph output widget in your frame
    a. In the INIT section of the code use the _get_object_size_ method to get the width & height of that widget in pixels
3. In the named section for the SAS/Graph Output widget you need to follow these steps:
    a. Call the _get_info_ method to get a list of information about where the user clicked
    b. Extract the x coordinate from that list
    c. Convert it to a percentage of the graphic area from the left using the following formula

        x_pct=(x-(width*x_offset))/(width*x_length)

        where x_pct ..... percentage across

the graphic area from the left
x ..... x coordinate in pixels from the list
width ..... width of the widget in pixels
x_offset ..... percentage that origin of axes is offset
x_length ..... percentage that x-axis is

d. Now we do the same for the y-axis: extract the y coordinate from that list
e. Convert it to a percentage of the graphic area from the left using the following formula – note that it is slightly different to the previous formula since the y coordinate is from the top down, whereas the x-coordinate is from left to right.

y_pct=((height-y)-(height*y_offset))/(height*y_length)

where y_pct ..... percentage across the graphic area from the left
y ..... y coordinate in pixels from the list
height ..... height of the widget in pixels
y_offset ..... percentage that origin of axes is offset
y_length ..... percentage that y-axis is

f. Now that we have the x & y percentages, we can use them in various ways. If we have dates on the x-axis then we could do the following, for example
   i. We could calculate the nearest date corresponding to where the user clicked with this code

date=round(x_pct*(maxdate-mindate)+mindate) ;

where date ..... date nearest to where user clicked
x_pct ..... percentage across the graphic area
maxdate..... maximum date on the x-axis
mindate ..... minimum date on the x-axis

   ii. We can apply a fuzz factor to gather dates +/- 5% from where the user clicked with this code

lowdate=round((x_pct-0.05)*(maxdate-mindate)+mindate) ;
highdate=round((x_pct+0.05)*(maxdate-mindate)+mindate) ;

where lowdate ..... date 5% less than where click was
highdate ..... date 5% more than where click was

a where statement could then apply this information as follows in order to select data matching this range …

rc=where(dsid,putn(lowdate,'5.')||'<=date<='||putn(highdate,'5.')) ;

Tips for extending the technique

- Values required for ORDER= on the AXIS statements can be calculated depending on the data. These can then easily be substituted into the SUBMITed code.
- Can use a formula to calculate at what points major and minor tickmarks should be put, and also what the maximum and minimum values of the axes should be so as to give nicely rounded steps
- You can also make use of fuzz factors for checking y-axis values, as in the example code
- Knowing the data that was clicked on allows all kinds of functions to then be implemented such as drill down, data pop-up, further selection to clarify requirements, etc.

Example code

```
* Define offset amounts in percent ;
%let xoff=15;
%let yoff=20;

* Define length of axes in percent ;
%let xlen=70;
%let ylen=70;

* Calculate the origin of the second y axis in case
we want to use it ;
%let y2origin=%sysevalf(&xoff+&xlen);

init:
 * Get size of the SAS/Graph Output widget ;
  call
notify('graph','_get_object_size_',width,height,'pix
els') ;

 * Use percentages defined in macro variables in SCL
variables to be used in submit block ;
  origin_x="&xoff pct" ;
  xaxis_length="&xlen pct" ;
  origin_y="&yoff pct" ;
  yaxis_length="&ylen pct" ;

 * Define the minimum and maximum date values on the
x-axis ;
  mindate='1jan2000'd;
  maxdate='31dec2000'd;

 * Generate the graph ;
  submit continue ;
   * Delete any old graphs that may be there ;
    proc datasets lib=work ;
      delete test / mt=catalog ;
    run ;
    quit ;

   * Generate some test data for graphing ;
    data a ;
      format x month. ;
      do z=1,2,3 ;
        do x=&mindate to &maxdate ;
          y=ranuni(1)*.36 ;
          y=y+(z-1)*.333 ;
          if y>1 then
```

```
                y=1 ;
          output ;
        end ;
      end ;
    run ;

   * Define and generate the graph - see paper for
details ;
    symbol1 i=join v=dot height=1 ;
    axis1 origin=(&origin_x,&origin_y)
          order=('1jan2000'd to '1jan2001'd by
month)
          length=&xaxis_length
          offset=(0,0)
          minor=none
          label=('Month') ;
    axis2 origin=(&origin_x,&origin_y)
          order=(0 to 1 by 0.1)
          length=&yaxis_length
          offset=(0,0)
          label=(' ') ;
    goptions nodisplay ;
    proc gplot data=a gout=work.test ;
      plot y*x=z / haxis=axis1
                   vaxis=axis2 ;
    run ;
    quit ;
    goptions display ;
  endsubmit ;

 * make list to hold information about where the
user clicked ;
  info=makelist() ;
return ;


term:
  info=dellist(info) ;
 * avoid compiler warnings ;
  rc=rc ;
return ;


graph:
 * get list of information about where user clicked
;
  call notify('graph','_get_info_',info) ;

 * extract x coordinate ;
  x=getnitemn(info,'x',1,1,0) ;
 * calculate thecorresponding percentage ;
  xpct=(x-(width*.&xoff))/(width*0.&xlen) ;

 * extract y coordinate ;
  y=getnitemn(info,'y',1,1,0) ;
 * reverse y-coordinates since they go from top to
bottom,
   but we need bottom to top ;
  y=height-y ;
 * calculate the corresponding percentage ;
  ypct=(y-(height*0.&yoff))/(height*0.&ylen) ;

 * Calculate what data click relates to ;
  roundx=round(xpct*(maxdate-mindate)+mindate) ;
 * use the x fuzz factor to get a range around the
click ;
  lowx=round((xpct-xfuzz/100)*(maxdate-
mindate)+mindate) ;
  highx=round((xpct+xfuzz/100)*(maxdate-
mindate)+mindate) ;

 * use the y fuzz factor to get a range around the
click ;
  lowy=ypct-yfuzz/100 ;
  highy=ypct+yfuzz/100 ;

 * open the dataset on which the graph was based ;
  dsid=open('a') ;
  call set(dsid) ;
 * select the data in the fuzzy square around the
click ;

rc=where(dsid,putn(lowx,'12.9')||'<=x<='||putn(highx
,'12.9')||' and '||

putn(lowy,'12.9')||'<=y<='||putn(highy,'12.9')) ;

 * put information about the points near the click
into a list ;
  popup=makelist() ;
  do while(fetch(dsid)=0) ;
    rc=insertc(popup,putn(x,'date9.')||' -
'||putn(y,'6.4'),-1) ;
  end ;

 * display the list ;
  rc=popmenu(popup) ;

 * clean up ;
  popup=dellist(popup) ;
  dsid=close(dsid) ;
return ;
```

Contact information

Philip Mason
Wood Street Consultants Ltd.
16 Wood Street, Wallingford, Oxfordshire, OX10 0AY,
England
Phone/Fax: (0771) 105449
Email: philipmason@email.com
Web: http://how.to/usesas