Generating Graphics using DSGI (Data Step Graphic Interface) Enables mouse selection of graphic points.

Bruce Kayton, Trilogy Consulting Corp.

The Problem

Looking at a graph and visibly seeing a problem or needing more information relating to specific data represented by the graph can be frustrating if one does not have the tools or expertise to do so. An experienced SAS® user can modify the code and regenerate the graph or identify the input data for further analysis without much problem. It is, however a challenge for the user that is merely using a predefined application to view a graph. It invariably means asking for a modification to the application or an analysis request of the data. With workloads and priorities the results are never immediate.

The Solution

Enable the user to generate the graph and using point and click operations with a mouse, select "problem" points or data points that need further clarification and exclude them from the graph or pull up more information relating to the data. All very well you, might say, but how?

This solution to the problem proposes that a graph be generated with an option to interactively select points on the graph for further analysis or exclusion. The user can generate "parameters" by selecting various points on the graph and then the dat represented at those points will be used to subset the data. A more comprehensive graph using SAS/GRAPH® can then be generated, if required, with any "problem" areas eliminated or clarified.

It would have been convenient to have the user select points on the graph produced by SAS/GRAPH® but unfortunately the only capability at present is to view data points on a GRAPH object within a FRAMES application nothing can be done with the data except to view it.

First generate the graph. For this I used DSGI. This method allowed the co-ordinates of the data points to be stored while maintaining a link to the actual data representing the points on the graph.

Follow the following steps to generate the graph:

1. Determine minimum and maximum values of each variable that is represented by each axis.

2. Calculate the number of intervals you need on each axis. eg An axis from 1 to 10 can have 9 intervals with an increment of 1 or 18 with 0.5. (In the application that I worked on I used 16 as the maximum number of intervals. Any more than that and the graph gets too difficult to read.

I used the following method to accomplish the task of calculating the intervals and the associated sub intervals.

Min_val and max_val are the minimum and maximum values of the data represented on the axis. I started with a step of 0.0001 and determined the number of intervals that there would be. If there would be more that 16 I multiplied the step by 5 until I reached an interval less than or equal to 16. Steps would have been 0.0001, 0.0005,0.001,0.005,0.01 etc Interval would be the value returned to the calling program (always <= 16). Int would be the increment returned and smallint would be the subincrements for plotting the scale on the axis.

```
**********************************************************************
   method min_val max_val interval int smallint 8;
    interval=16;int=0.0001;smallint=0.25;rep=0;
    if min_val=max_val then do;
      interval=0;
    end;
    do while (interval ge 16);
      if interval ge 16 then do;
        rep+1;
        if mod(rep,2)=0 then do;
          smallint=.25;
          int=int*2;
        end;
        else do;
          smallint=.20;
          int=int*5;
        end;
      end;
      min_val=min_val-mod(min_val,int);
      if round(max_val,int) ne max_val then
        max_val=max_val+(int-mod(max_val,int));
      interval=((max_val-min_val)/int);
      interval=int(interval+0.9999);
   end;
**********************************************************************
```

3. Draw each axis line and plot the scale on each axis. Xaxmax and xaxmin are the maximum and minimum co-ordinates in the window available for the graph. Test if min and max values are the same, in which case there will only be 1 point in the center of the graph. Use only 96% of the area available to allow for annotation alongside the y-axis.

```
*****************************************************************
   xwidth=xaxmax-xaxmin;
   if x_min eq x_max then do;
     xinc=xwidth;
     x=xaxmin+(0.5*xwidth);
     int=1;
   end;
   else do;
     xinc=(xwidth*.96)/interval;
     x=xaxmin+(.02*xwidth);
   end;
   t_val=x_min;
   do i=x to xaxmax  by xinc;
     a=left(put(t_val,best.));
     rc=gdraw('TEXT',i,yrange*.23,a);
     rc=gdraw('LINE',2,i,i,yrange*.235,yaxmin);
     if i+xinc le xaxmax then do;
       do z=i to i+xinc by xinc*smallint;
         rc=gdraw('line',2,z,z,yrange*.245,yaxmin);
       end;
     end;
     t_val=t_val+int;
   end;
*****************************************************************
```

4. Plot the data values on the graph. At the same time store the co-ordinates along with the data values in an SCL list for data value identification when clicked on at a later stage. In this case time_x is the data being plotted on the x-axis and conc_y is the data being plotted on the y-axis. Each x and y co-ordinate is calculated according to where it would fit on the scale of axis's already plotted.

```
*****************************************************************

x_cord=(xaxmin+(.02*xwidth))+
            (((time_x-x_min)/(x_max-x_min)) * (xwidth*.96));
y_cord=(yaxmin+(.02*yheight))+
            (((conc_y-y_min)/(y_max-y_min)) * (yheight*.96));
rc=gdraw('mark',1,x_cord,y_cord);

*************************************************************
* calculates the x and y co-ordinates as they would be
* determined on the FRAME object NB:: (0,0) is top left corner
* of object and (0,0) is bottom left corner of graph
*************************************************************
x=int((x_cord/xrange)*obj_wdth);
y=int(((yrange-y_cord)/100)*obj_hgt);
*************************************************************
* Inserts data values into list with a name of x and y
*  co-ordinates concatenated;
*************************************************************
plotlist=insertn(plotlist,time_x,-1,x!!' '!!y);
plotlist=insertn(plotlist,conc_y,-1,'CONC');
plotlist=insertn(plotlist,y_cord,-1,'Y');
plotlist=insertn(plotlist,x_cord,-1,'X');
```

You will notice that secondary co-ordinates are stored in the list (y_cord and x_cord). This is because the top left co-ordinates of a SAS/GRAPH® object within a frame are assigned (0,0) co-ordinates and DSGI graphics assigns (0,0) co-ordinates to the bottom left hand corner co-ordinates.

Now that the graph has been generated it can be presented within the FRAMES application and the user can click on points on the graph. To do this the generated graph needs to be saved in a standard graphic format for presentation within a frames object. In this application I allow a user to click on a point within a radius of 4 pixels around each graph point. The user can then be presented with an option to cancel the selection or exclude the point from the graph. The graph is then regenerated to reflect the modification. Of course, any action can be performed using the identified data value(s).

```
*****************************************************************
* _get_info_ method retrieves x and y co-ordinates from object
*  that has been clicked on and puts results in an internal list.
* Co-ordinates are named items in the list and are retrieved
* accordingly.
*****************************************************************
call notify('graph','_get_info_',listinfo);
    x     =getnitemn(listinfo,'X');
    y     =getnitemn(listinfo,'Y');
    do i=x-4 to x+4;
      do i2=y-4 to y+4;
        ***************************************************
        * Search around the co-ordinates clicked upon for a
        * match against the stored co-ordinates on the graph
        ***************************************************
        idx=nameditem(plotlist,i!!' '!!i2);
        if idx then do;
          ***************************************************
          * Positive match! Retrieve data values from list.
          ***************************************************
          time_val=getitemn(plotlist,idx);
          conc_val=getnitemn(plotlist,'CONC',1,idx);
          y_cord=getnitemn(plotlist,'Y',1,idx);
          x_cord=getnitemn(plotlist,'X',1,idx);

        ***************************************************
        * At this point the data values time_val and conc_val
        * are the data values represented on the graph.
        * They can be used in further options, for drill down,
        * exclusion, specific calculations, menu options etc etc.
        ***************************************************
          end;
        end;
    end;
    return;
```

While the user is selecting points on the graph the application builds an SCL List of the points to be excluded from the final graph output. These canl be used for further analysis or to subset the data when the graph is reproduced using SAS/GRAPH®.

Author

Bruce Kayton
Trilogy Consulting Corporation
850 S Green Bay Rd
Waukegan, IL 60085-9905
(800) 323-7528
71213,2714@compuserve.com

Acknowledgment

The author wishes to thank Andy Norton of Trilogy Consulting Corp. Kalamazoo, MI for his suggestion to make use of DSGI and for his valuable input into finding a reasonable solution to this problem.

REFERENCES

1. SAS/GRAPH Software: Reference, Version 6, First Edition, volumes 1 and 2

SAS®, SAS/GRAPH® are registered trademarks of SAS Institute, Inc., Cary, North Carolina, in the USA and other countries. ® indicates USA registration.