Using the Annotate = Data Set to Highlight the Graphical Quality of the G3D Procedure

Kevin M. Hillstrom
The Garst Seed Company®
A Division of ICI Seeds®

Abstract

Annotate = data set can be used along with PROC G3D to enhance the visual quality of three dimensional graphics. Colors can be mapped onto the grid to represent countour levels, and annotate can place state boundaries on top of the contour to improve visual quality of the graphic.

Introduction

The Research Information Systems (RIS) section of the Garst Seed Company® uses SAS/GRAPH® software to present agricultural research data in a clear, concise manner to our plant breeders, bioscientists, sales managers and agronomists. During the last year our department has used SAS-GRAPH more extensively, as our expertise has grown and comfort with the package has increased. It was during this time that I began to play around a little with PROC G3D, a procedure that creates three dimensional graphics. My goal was to create a three dimensional graph of a state, in this case Iowa, and map precipitation values to the graph in such a way that areas of high precipitation would appear to be hills on the graph, while areas with little precipitation would look like valleys. I had hoped to use different color levels to represent contour levels on the graph. After a short time, I realized that the procedure only allows one color to be mapped onto the graph. This paper will demonstrate how the Annotate = data set can be used to accomplish this.

The Procedure

The program can be divided into four separate sections.
1) Creating the grid that PROC G3D uses.
2) Use the Annotate = data set to assign values for the state border that will be drawn on the graph.
3) Use the Annotate = data set to assign colors that depict the relative height differences on the z axis.
4) Plot the graph.

I began by reading in all pertinent data sets. I created a data set that contained all data points for the border of Iowa, a data set that had the x and y coordinates for each city in Iowa that I had precipitation for, and finally a data set that had the precipitation data. I merged the precipitation data with its appropriate coordinates, and began the procedure.

PROC SUMMARY was used on the data set that contained the border coordinates for Iowa to find the maximum and minimum x and y values. At that point I read in the output data set, and added a value of 1.4*(highest x value minus lowest x value)/(number of grids desired for x axis, in my case, 30). This is subtracted from the lowest x value as well. I found that PROC G3GRID, which will be used later, often did not include all points on the extremes of the graph. This accomplishes that. The values are then converted to macro variables that are plugged in later on. The code used is given below.

```
proc summary missing nway data=borders;
var x y;
output out=limits
min=lowx lowy max=highx highy;

data limits; set limits;
high=highx + 1.4*(highx-lowx)/30;
low=lowx - 1.4*(highx-lowx)/30;
highx=high; lowx=low;
rany=(highy-lowy)/20;
ranx=(highx-lowx)/30;
call symput('lx',lowx);
call symput('hx',highx);
call symput('ly',lowy);
call symput('hy',highy);
call symput('ry',rany);
call symput('rx',ranx);
```

PROC G3GRID is now used to create an even grid of data points. Macro variables are used from above to determine the range of the graph.

```
proc g3grid data=precip out=precip;
grid y*x=rain /
axis2=&lx to &hx by &rx
axis1=&ly to &hy by &ry noscale;
run;
```

Now that we have a grid to create a three dimensional graph from, annotate = data set can be used to do several things. First, we want to assign z axis (precipitation) values to the boundaries of the state. The code described below does just that.

```
data anno; set borders precip;
    proc sort; by x y; run;
data anno; set anno; by x y;
retain dum 0; newx=x;
if state=. then dum=x;
else if state ^=. then newx=dum;
return; drop dum;
    proc sort; by newx y;
```

What was done above is basically this. We look for a value of state that is not equal to missing. This happens only in the input border data set, since only x and y values exist. If that is the case, we create a newx value that is equal to the last observation that had a precipitation value. The following mini-data set shows this.

rain	x	y	newx	state
4	1	2	1	.
3	1	3	1	.
.	1.2	2.6	1	19
3	2	2	2	.
6	2	3	2	.
.	2.6	3	2	19

As you can see, when state is not equal to missing, a value for newx is assigned equal to the value of x in the previous observation.

Next, the data set is sorted by the newx value, and the y. An approximate value for the rain variable for border data points is then assigned as follows:

```
data anno; set anno; by newx y;
    retain dum 0;
    if rain ^ =. then dum=rain;
    else if rain=. then rain=dum;
    return; drop dum;
```

In this step, any time the rain variable comes up missing (only in the state border data set), a value of the z variable, in this case rain, is assigned equal to the last non-missing rain value. In all cases, this will be equal to the nearest prior rain value when passing through the data set. An example is shown below.

rain	x	y	newx	state
4	1	2	1	.
4	1.2	2.6	1	19
3	1	3	1	.
3	2	2	2	.
6	2	3	2	.
6	2.6	3	2	19

Rain values for the borders of the state are shown in bold print. Finally, the state border points can be subset. They are sorted by a variable that I called order, which is just the original order they were input at the start of the program. The code below assigns the correct annotate = data set variables to draw a black line on the graph to represent the state borders.

```
data anno; set anno;
    if state ^ =.;
        proc sort; by order;
data anno; set anno;
    if order=1 then function='poly    ';
```

We are now ready to tackle the third step, assigning colors to the contour levels we desire. This is accomplished by creating necessary annotate = data set variables, and then sorting by x and y. Now, the data set is passed through by each level of x. As y ascends, we draw a line from data point to data point in the color of the desired contour.

```
data anno1; set precip;
    when='a'; size=1; position='5';
    xsys='2'; ysys='2'; zsys='2'; z=rain;
        proc sort; by x y; run;
data anno1; set anno1; by x;
    if first.x then function='move    ';
    else function='draw    ';
    if rain<3.5 then color='blue    ';
    else if 3.5<=rain<4.5 then color='green    ';
    else if 4.5<=rain<5.5 then color='red    ';
    else color='purple    ';
        proc sort data=anno1 out=anno2;
         by y x; run;
```

In the last line, the anno1 data set is sorted by y and then x, and output as a new data set. We will pass through this data set just as we did above, an assign values for vertical lines on the graph.

```
data anno2; set anno2; by y;
    if first.y then function='move    ';
    else function='draw    ';
    if rain<3.5 then color='blue    ';
    else if 3.5<=rain<4.5 then color='green    ';
    else if 4.5<=rain<5.5 then color='red    ';
    else color='purple    ';
```

Finally, we can begin step 4, plotting the graph. We have to set together all annotate data sets, and plot the graph.

```
data anno; set anno1 anno2 anno; run;
        proc g3d anno=anno data=precip;
        plot y*x=rain /
        ctop=white noaxes zmin=0 zmax=8
        nolabel rotate=10 tilt=30;
        title f=complex h=2 c=black
        'Iowa rainfall totals during the last 60 days'; run;
```

A tilt of 30 degrees was used to show three dimensional aspect as much as possible, while distorting the state boundary lines as little as possible. The ctop = white option was used for a couple of reasons. First, I assigned a white backround, so in essence, the real plot is being plotted in the background color. Then, the annotate = data set option writes over the top of the graph in the colors desired.

Legends and tables can be created and placed on the graph using PROC GREPLAY.

Conclusion

Using the annotate = data set facility of SAS/GRAPH® greatly enhances the visual output of PROC G3D. State borders are easily placed on the grid, and colors can be added to the grid to represent different rainfall regimes.

® Trademark Acknowledgements

References

SAS Institute Inc. SAS® User's Guide: Basics, Version 5 Edition. Cary, NC: SAS Institute Inc., 1985. 1290 pp.

SAS Institute Inc. SAS/GRAPH® Guide for Personal Computers, Version 6 Edition. Cary, NC: SAS Institute Inc., 1987. 534 pp.

Other Acknowledgements

The author would like to thank David Jacobs and Shaw Ling Wang for their technical support; Jean Cormack, Mary Fisher, Diane Hamilton, Aliene Jensen, Deb Ketcham, Ron Mowers, Mary Olsem, the rest of RIS, and my wife Tori Benz-Hillstrom for their creative assistance.