

Creating Maps in SAS/GRAPH

By Jeffery D. Gilbert, Trilogy Consulting Corporation, Kalamazoo, MI

Abstract

This paper will give an introduction to creating graphs using the PROC GMAP procedure in SAS/GRAPH. Instruction will be given to create basic maps using map tables provided by The SAS Institute. In addition, some special topics will be discussed, including SAS/GRAPH global statements and the annotate facility when using PROC GMAP.

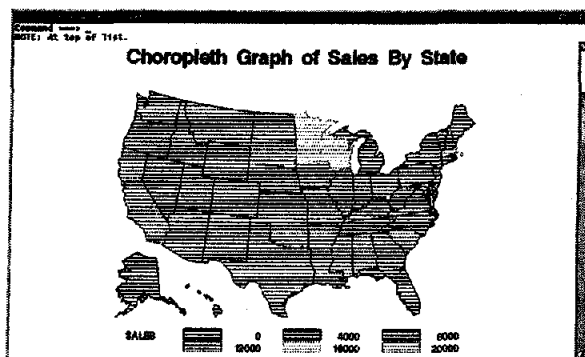
Maps in The SAS System

Maps can be used for many kinds of reports, and are very useful for visually describing data across a geography, such as the United States.

The SAS System has three basic types of maps:

- 1) The CHOROPLETH (or CHORO for short) map, which displays data on a flat map.
- 2) The PRISM, which shows maps three-dimensionally, and can give a greater feel for differences and between states/regions.
- 3) The SURFACE map, which shows more general areas and can be useful for picking out clusters.

A basic CHORO map that many people might want to produce looks like the below picture:



The code which produces this graph is:

```
options reset=global;
options gunit=pct cback=white htitle=6 htext=3
ftext=swissb ctext=blue;

%macro st2reg;
/* Macro to flip state into regional SAS User
Groups. */
length region $10;
if state in (17 18 19 20 26 27 29 31
            38 39 46 55)
    then region='MWSUG';
else if state in (02 16 41 53)
    then region='PNYSUG';
else if state in (05 22 40 48)
    then region='SCSUG';
else if state in (01 12 13 21 28 37 45
                47 51 54 72)
    then region='SESUG';
else if state in (09 10 11 23 24 25 33
                34 36 42 44 50)
    then region='NBSUG';
else if state in (04 06 08 15 32 49)
    then region='WUSS';
else region='UNKNOWN';
%mend;

* Create random "sales" value for each state.;
proc sort data=maps.us out=sls(keep=state)
nodupkey;
by state;
run;

data sls;
set sls;
statnm = fipname(state);
%st2reg;
sales=round(abs(10000*normal(1)),1);
run;

title 'Choropleth Graph of Sales By State';
proc gmap all map=maps.us data=sls;
id state;
choro sales;
run; quit;
```

This program contains several parts:

- 1) Define the SAS/GRAPH options.
- 2) Define a macro to flip state into Regional SAS Users Groups.
- 3) Create a dataset containing random sales volume for the fictitious company within each state/region. For purposes of this example, sales volume is simply a random normally distributed variable.
- 4) Run the graph using a discrete ID variable and CHORO variable – sales volume. Note that two tables must be specified:
 - a) a MAP= table containing the x and y coordinates for graphing, and
 - b) DATA= table containing the values, such as sales volume, to be graphed.

The ID variable is used by the procedure to tie the two tables together.

The MAP= Table

SAS/GRAPH comes with numerous map tables pre-defined. These tables are found in the MAPS library, though not all are immediately useful to a procedure. The ones that are immediately useful will contain certain variables at a minimum:

- 1) x – the x-axis coordinate.
- 2) y – the y axis coordinate.
- 3) Segment – referencing geographic boundaries.
- 4) Id or state – which references a discrete state indicator.

In the MAPS library supplied with The SAS System, there are a number of datasets which do not include these variables. These are useful for other purposes, such as using the ANNOTATE facility. Always use caution when picking the dataset you plan to use.

The DATA= Table

The data table is created by the programmer. It must have a key variable with the same name as the key variable on the MAPS= table. For example, if creating a graph of the United States, both the MAPS= and DATA= tables must contain the variable state, with the values matching up as well. In other words, if the MAPS= table contains the value of '26' for the State of Michigan, then the DATA= tables must also contain a value of '26' for the State of Michigan.

GLOBAL Statement: PATTERN

Up until now, no SAS/GRAPH global statements have been specified. The maps look pretty bland, and could be spruced up a bit. After all, it is difficult to tell state borders, particularly on black and white output as on this paper. In the next example, seven pattern statements are specified. These take the following syntax:

- 1) PATTERN<x>, where x is the pattern

number (up to 20). One pattern will be used for each breakout of the data – if there are seven regions (as in the case of the upcoming example), then seven PATTERN statements should be specified. Otherwise, a default pattern will be used, which can be undesired.

- 2) V=m<y><c><z>, where:

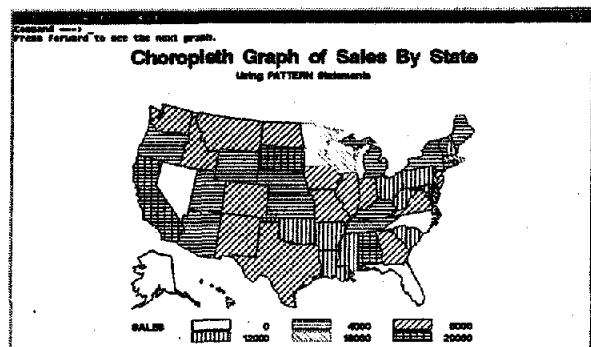
- a) y = an integer specifying how many lines are to be drawn,
- b) c = either N or X. An N specifies that the lines in the pattern are to be drawn at an angle, and X specifies that the lines are to be criss-crossed, and
- c) z = an integer which specifies the angle at which the lines are drawn.

- 3) C=<color> to specify a color.

For example, the following pattern statements, when applied to the CHORO map produced earlier:

```
pattern1 v= c=black;
pattern2 v=m1n00 c=pink;
pattern3 v=m2n45 c=green;
pattern4 v=m3n90 c=pink;
pattern5 v=m4n135 c=cyan;
pattern6 v=m1x00 c=brown;
pattern7 v=m2x45 c=magenta;
```

Results in the following map:

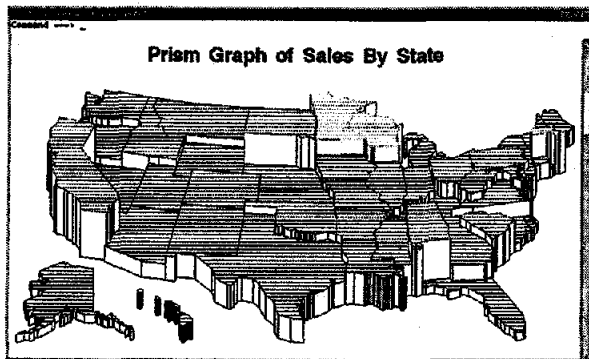


On color or black and white output, the PATTERN global statement greatly increases the readability of this map.

Map Types

As mentioned previously, there are three different types of maps that The SAS System can produce: CHOROPLETH (CHORO), PRISM, and SURFACE. This section will give basic examples of the PRISM and SURFACE maps.

A PRISM map can be created similarly to the CHORO map:

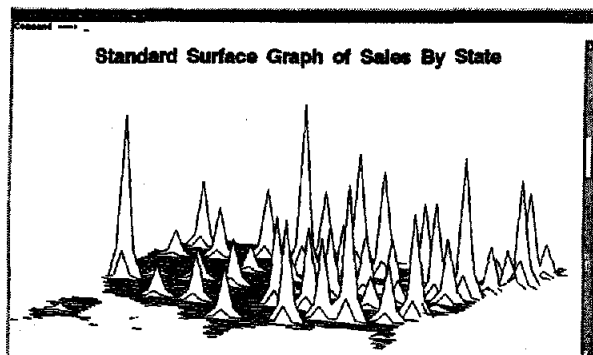


This graph was created using the following code:

```
* Prism Map;  
title 'Prism Graph of Sales By State';  
proc gmap all map=maps.us data=sls;  
  id state;  
  prism sales / nolegend;  
run; quit;
```

Again, an ID variable was needed to tie together the MAP= and DATA= tables, and this time specified PRISM. The NOLEGEND option simply specifies that, even though the procedure groups the levels of sales for us, it will not display the legend. Because the map is three dimensional, it is clearer how a state shapes up for sales.

A SURFACE graph is shown below:



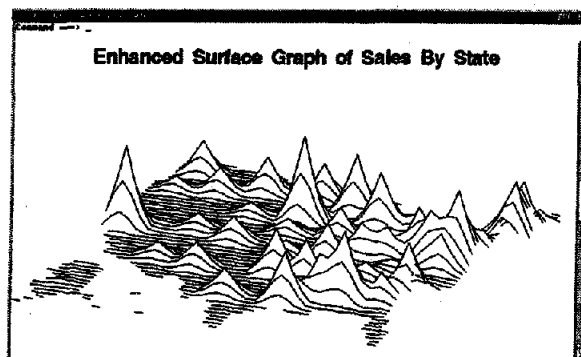
The code so far has been very similar for each type of map, as you can see.

```
* Standard Surface map;  
title 'Standard Surface Graph of Sales By State';  
proc gmap all map=maps.us data=sls;  
  id state;  
  surface sales;  
run; quit;
```

This surface map shows how sales are shaping up, and is stronger at showing regional clusters. However, it is not useful for viewing particular states.

Surface graphs contain a couple of options for making the view of this type of graph a bit easier to read. :

- 1) The option TILT will allow you to tilt the map to put it at an angle easier to read. Thirty degrees seems to be a pretty readable angle.
- 2) The CONSTANT option will help spread the base of the spikes so that clusters are easier to identify. Fifty seems to be helpful in picking out clusters in the example below.



The code which produced this map is:

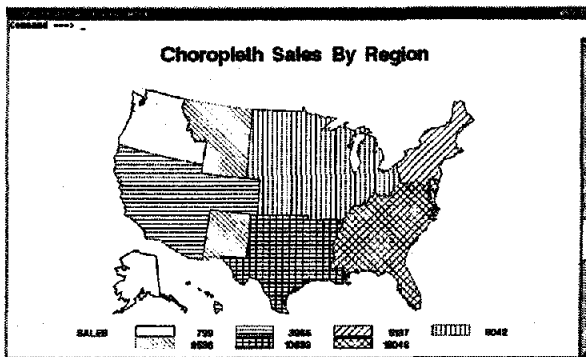
```
* Enhanced Surface map;  
title 'Enhanced Surface Graph of Sales By State';  
proc gmap all map=maps.us data=sls;  
  id state;  
  surface sales / constant=50 tilt=30;  
run; quit;
```

Removing Borders for a Region

Often, a particular region is desired to be graphed rather than by state, as supplied by

The SAS Institute. For example, if we wanted to look at the (fictitious) sales to Regional SAS Users' groups, not by state, we need some way to remove the lines and plot just the desired borders. Thankfully, that option is available with PROC GREMOVE.

To use this procedure, start with the desired MAP= table, and create a new variable which is transformed from the ID variable on that table. Then run it through PROC GREMOVE to correctly remove the state borders, as in this example:



The GREMOVE procedure created the appropriate boundaries using this program code:

```
* Use GREMOVE to change the borders to match
SAS Users Groups.
proc summary data=sas nway missing;
  class state region;
  var sales;
  output out=sas_reg(drop=_type_ _freq_) sum=;
run;

data maps(compress=NO);
  set maps.us;
  tset2reg;
run;
proc sort data=maps;
  by region;
run;

proc gremove data=maps out=map_reg;
  by region;
  id state;
run; quit;

proc gmap all map=map_reg data=sas_reg;
  id region;
  choro sales / discrete;
run; quit;
```

In the PROC GREMOVE, two statements are needed:

- 1) the ID variable, to tell the procedure the name of the old key variable, and
- 2) the BY variable is the name of the new key on the new MAP= table.

CAUTION: PROC GREMOVE cannot use tables which are compressed. A good habit to get into, when using PROC GREMOVE, is to specify that the table not be compressed.

Just Maps...

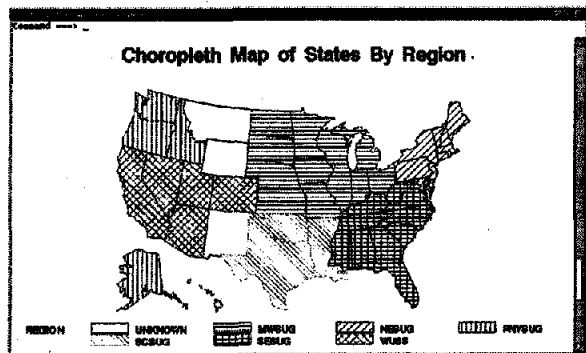
Sometimes, just a map is desired rather than plotting continuous variables, such as sales volume.. This can be done using the DISCRETE option on the CHORO statement, which would indicate that the "CHORO" variable listed is a discrete variable.

As an example, state borders can be mapped within the Regional SAS Users Groups areas using GREMOVE and doing the following:

```
proc gremove data=maps out=map_reg;
  * NOTE: Both tables were created earlier.;
  by region state;
  id state;
run; quit;

title 'Choropleth Map of States By Region';
proc gmap all map=map_reg data=sas_reg;
  id state;
  choro region / discrete;
run; quit;
```

This will result in a map such as:



Using Annotations on a Map

The annotate facility is a useful, though sometimes tricky, way to add meaning to a map. Often it will trial and error to make annotations fall in the correct spot. While the objective of this paper is not to teach the annotate facility, it would be irresponsible to exclude at least an example of using it for customizing a graph.

A good resource for learning the annotate facility is listed in the bibliography.

While the annotate facility is useful for

annotating CHORO maps, it can get tricky when attempting to annotate PRISM maps, and very confusing when trying to annotate SURFACE maps.

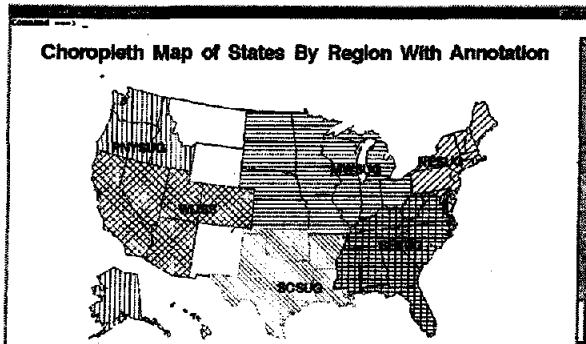
An example of annotating a CHORO map, for the purpose of adding Regions SAS Users Groups names to the map, is below:

```
* Create the ANNOTATE dataset.;
proc summary data=map_reg(keep=region x y)
    nway missing;
    class region;
    var x y;
    output out=annotate(drop=_type _freq) max=;
run;
data annotate;
    set annotate end=end;
    retain position '5' xsys '2' ysys '2'
        hsys '4' function 'LABEL'
        style 'SWISSB' size 1 color 'BLACK';
    if not index(region, 'UNKNOWN');
    y = y-.1; x=x-.1;

    text=region;
run;

title 'Choropleth Map of States By Region With
Annotation';
proc gmap all map=map_reg data=sls_reg;
    id state;
    choro region / discrete annotate=annotate
        nolegend;
run; quit;
```

This will result in the following map:



Cleaning Up

Because mapping in the SAS System is so powerful, with many options not mentioned here, trial and error will be a method used often to make maps look as desired. Creating many maps can create some problems with disk space and memory, so it is strongly suggested that maps (and graphs) be deleted if no longer needed. The following procedure can be used to clean up all graphs in the default catalog:

```
* Clean up the maps you have created.;
* USE WITH CAUTION!!!;
proc greplay nofs igout=work.gseg;
    * delete _all_;
run; quit;
```

Of course, using a statement like DELETE GMAP; will delete only the map named GMAP on this catalog. Use caution when using the DELETE _ALL_; statement in PROC GREPLAY, as all maps and other graphs created by SAS/GRAPH will be deleted.

Summary

The mapping procedure in The SAS System is a powerful tool and can be customized to produce several different kinds of maps. Because of the power offered by The SAS System, care must be taken to understand the inputs into a map to result in the output desired. Often, the process of producing a map will require some trial and error. But the ability to display a graph within The SAS System is a powerful and meaningful to display information graphically.

Reference and Further Study

- 1) SAS/GRAPH Software, Reference, Volume 2, Chapter 29 (pages 1001-1062).
- 2) Replaying Graphics with PROC GREPLAY, by Jeffery D. Gilbert, SAS Institute, Proceedings of the Twenty-Fifth Annual SAS Users Group International Conference (pages 873-875)
- 3) Customizing SAS Graphs Using the Annotate Facility and Global Statements, by Jeffery D. Gilbert, SAS Institute, Proceedings of the Twenty-Fourth Annual SAS Users Group International Conference (pages 1002-1005)

Contact

Jeffery D. Gilbert (Jeff)
 Trilogy Consulting Corporation
 5278 Lovers Lane
 Kalamazoo, MI 49002
 Work Phone: (616) 344-4100
 Email: JefferyGilbert@hotmail.com