

Paper SIB-110

Moving from Listing Reports to Visual Reports using SAS/GRAPH® Maps

Elizabeth Ceranowski, SAS®, Cary, NC

ABSTRACT

Listing Reports, like the output from the Print Procedure, can be good. Summary reports, like those produced by the Report procedure, can be better. However, a good visual report, like a color coded map, can sometimes be more effective in communicating information. Don't have SAS/GIS®? You can do this using SAS/GRAPH®! This presentation will get you started in using the GMAP procedure to create graphical output and using ODS to publish the output. Attendees will also see examples of use, and basic syntax to get started with SAS/GRAPH® Maps and ODS. Code and output will be shown, as well as ways to customize the output.

The content of this paper can also be easily extended to be used as Stored Processes in the SAS Business Intelligence Platform, and could easily be added to a SAS Information Delivery Portal.

INTRODUCTION

This paper will walk you through producing a map from a SAS data set. This paper will focus on producing a US map, however, any country could be produced to align with your data. In addition, more detail will be added to the map to make it more useable colors will be set, labels will be added to the states, and options will be added to save the map as a GIF file. Further, a single state will be examined, and detail will be shown at the county level.

There are certainly many ways to do this in SAS, the focus of this paper is to help beginner and intermediate SAS users who might be unfamiliar with SAS/GRAPH started creating graphs.

THE DATA AND THE EXAMPLES USED IN THIS PAPER

The data used in this paper is entirely fictitious. Orion Star Sports & Outdoors is an international retail company that sells sports and outdoor products. The headquarters is based in the United States, and retail stores are situated in a number of other countries including Belgium, Holland, Germany, the United Kingdom, Denmark, France, Italy, Spain, and Australia.

Five people work for Orion Star selling collegiate branded merchandise to certain territories and select colleges and universities in the United States. Each person works in several different states. Their manager keeps up with their assignments using SAS listing and summary reports, but would rather have a map with color coded shading to make it quicker to locate who is working in a particular area.

The data set **ORION.SITES** contains 50 rows, one for each state, and two variables. **RegionRep** is a character variable containing a value 1 through 5. (Each sales representative has their own number.) **StCode** is the two character abbreviation for the state. Again, this data is entirely fictional.

LISTING REPORTS

A proc print step could be submitted to display this information, all 52 rows.

```
title1 'Orion Star Sales Rep Territories';
title2 'Fall 2008';
proc print data=orion.sites noobs;
  var Stcode RegionRep;
  format regionrep $territory.;
run;
```

This code produces the following partial output:

**Orion Star Sales Rep Territories
Fall 2008**

stcode	Region Rep
AK	Pat
AL	Pat
AR	Pat
AZ	Bruce
CA	Cassie
CO	Bruce
CT	Cassie
DE	Pat
FL	Ben
GA	Ben
HI	Cassie
IA	Pat
ID	Bree

SUMMARY REPORT

A proc report step could be submitted to display this information in a summary report, all 52 rows.

```
proc report data=orion.sites nowd;  
  column RegionRep Stcode;  
  define regionrep / group format=$territory.;  
  define Stcode / order width=6;  
run;
```

This code produces the following partial output:

**Orion Star Sales Rep Territories
Fall 2008**

Region	
Rep	stcode
Ben	FL
	GA
	MA
	MD
	NC
	PA
	TN
	VA
Bree	ID
	IN
	MT
	ND
	SD
Bruce	AZ
	CO
	KS

Both of these outputs are useful, but a geographic representation might be more useful.

USING A DATA STEP AND RETAIN

One easy way to list the data would be to create a dataset with one row per person, and have a character string with each state separated by a comma. Although this might be easier to reference, it will still take some time to find the information.

Orion Star Sales Rep Territories Fall 2008

RegionRep	States
Cassie	CA,CT,HI,IL,ME,MI,MN,MO,NH,NJ,NY,RI,VT,WV
Ben	FL,GA,MA,MD,NC,PA,TN,VA
Pat	AK,AL,AR,DE,IA,KY,LA,MS,OH,OK,OR,SC,TX,WA,WI
Bree	ID,IN,MT,ND,SD
Bruce	AZ,CO,KS,NE,NM,NV,UT,WY

Source Code:

```
proc sort data=orion.sites out=sites;
  by regionrep stcode;
run;
data rep_sites(keep= regionrep states);
  set sites;
  length States $ 60;
  by regionrep;
  retain states; * Contents of states will not be cleared at top of step;

  if first.regionrep then
    states=stcode;
    *This is first row for rep- initialize states to stcode;
  else
    states = trim(states) || ',' || stcode;
    * Concatenate the new states string with a comma followed by stcode
    Trim the States string to avoid blanks inside string;

  if last.regionrep;
run;
title 'Orion Star Sales Rep Territories';
title2 'Fall 2008';
proc print data=rep_sites noobs;
  format regionrep $territory10.;
run;
```

TYPES OF SAS/GRAPH OUTPUT MAPS

There are four types of maps that can be created using the GMAP procedure. The GMAP procedure produces two-dimensional (choropleth) or three-dimensional (block, prism, and surface) color maps that show variations of a variable value with respect to a geographic area.

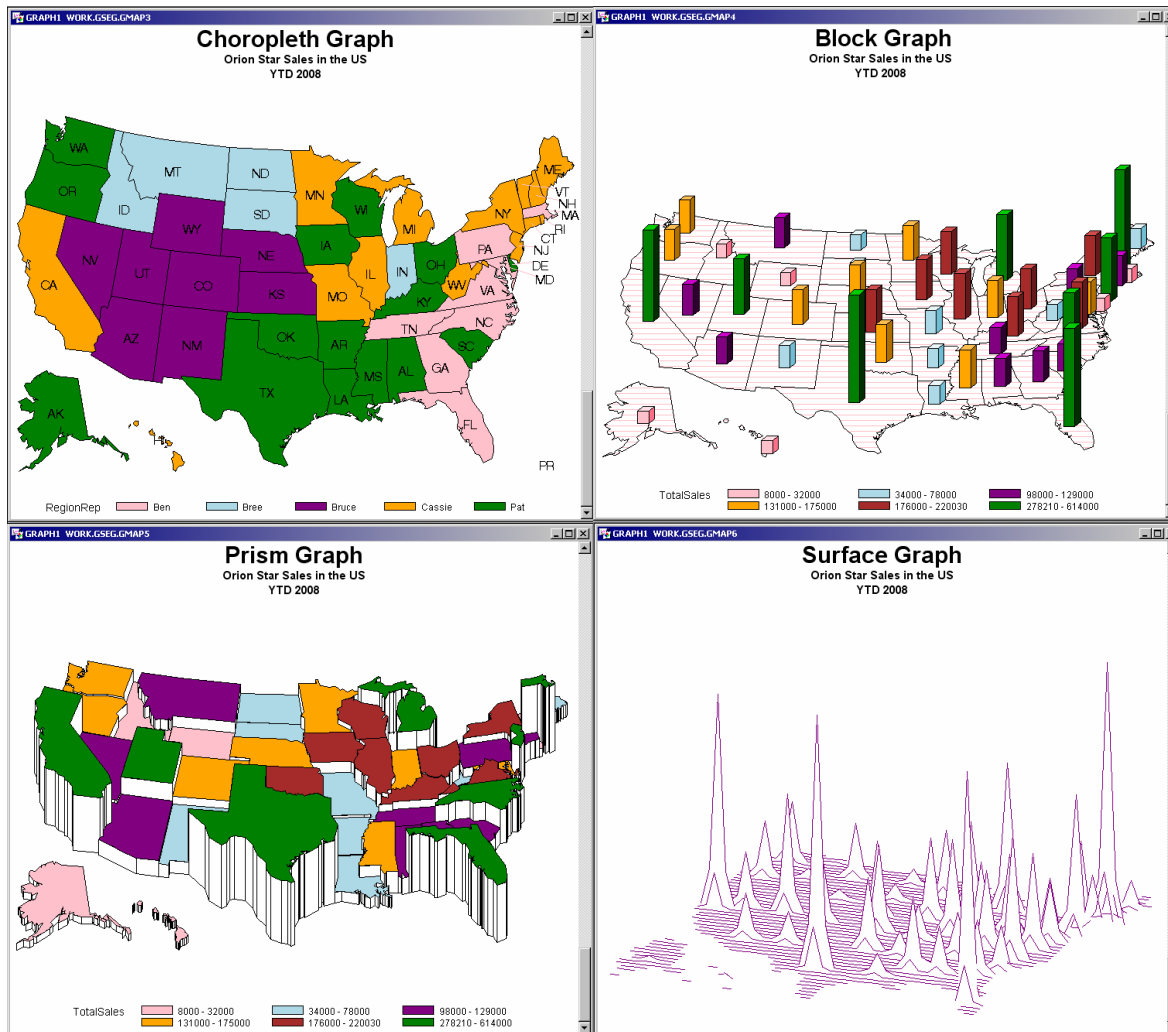
Two-dimensional (choropleth) maps indicate levels of the corresponding response variable by filling map areas with different colors and patterns. In the example for this paper, we will create a choropleth map with a coloring for each sales representative.

Block maps contain a block in the map for each boundary (in this example, a block for each state) where the height of the block is determined from a variable in the data. If there was a column for Total Sales for each state, for instance, the height of the bar could represent this value.

Prism maps will lift the region (in this example, each state) depending on the value of the response variable- the higher the region, the larger the value. If the Total Sales column was again used, the higher the value of sales, the higher the state would appear.

Surface maps display a spike at the approximate center of each map area to convey information about response variable values, like the previously mentioned Total Sales.

Examples of these graphs are shown below:



USING SAS/GRAPH MAPS

When creating a map using SAS/GRAPH, you need two data sets for input. The first data set contains the data you wish to map, called the response variables. In this case, the **ORION.SITES** data set contains the sales territory data and is the data that will be shaded on the map. The second data set that is needed contains the information about the geographic boundaries of the states. This information is provided in a maps library.

OBTAINING MAPS FROM SAS

The maps library is shipped with SAS/GRAPH. This library contains almost 300 SAS/GRAPH data sets of map data from around the world. The **MAPS.METAMAPS** data set contains information about the data sets in the Maps library. This data set includes information such as:

- Names of data sets used to map each country.
- Variable names and their labels in each data set.
- An indicator whether the data set is for use with SAS/GRAPH or for use only with SAS/GIS software.

Since this example focuses on creating a map of the United States, the **MAPS.US** data set will be used in this paper.

PREPARING THE DATA

The first thing that needs to be obtained is the FIPS (Federal Information Processing Standards) code for the state. The GMAP procedure will use this code to draw the map. This is very easy, using SAS functions. There are two functions that will create a FIPS code. The **STFIPS** function converts a state abbreviation to a FIPS state code. The **ZIPFIPS** function converts zip codes to FIPS codes. Since the data presented in this example has the state abbreviation, the **STFIPS** function will be used. This function requires one parameter, a two character state abbreviation contained in parenthesis. Note that the two character abbreviation can be mixed case. The **STFIPS** function ignores trailing blanks, but generates an error if the expression contains leading blanks.

```
data orion.sites;
  set orion.sites;
  state=stfips(stcode);
run;
```

Partial Output of the new **ORION.SITES** data set:

stcode	Region	
	Rep	state
AK	3	2
AL	3	1
AR	3	5
AZ	5	4
CA	1	6
CO	5	8
CT	1	9

CREATING THE REPRESENTATIVE FORMAT

Assigning names to the **RegionRep** code is done by using the **FORMAT** procedure. This method was chosen for two reasons: less data storage is needed for one character, and if someone leaves the company, the new hire name would only need to be changed in one place- the format, which is easier to change than the data.

Example code:

```
proc format;
  value $territory 1= 'Cassie'
                  2= 'Ben'
                  3= 'Pat'
                  4= 'Bree'
                  5= 'Bruce';
run;
```

This format procedure should be placed at the top of the program so it can be used below without problems.

PRODUCING THE MAP

The GMAP Procedure will be used to create the map. The basic syntax for the procedure is:

```
proc gmap map= <data set from maps library containing geographic data>
  data= <data set containing values to plot, called response variables>;
  id < variable containing map unit>;
  <Choro|Prism|Block|Surface> <response variable - value to plot>;
run;
```

Note that in this paper the <> signifies that the user should replace the <> symbols and the words inside. Also, a vertical bar- '|' represents "OR". In the syntax above, you must choose a CHORO statement, a PRISM statement, a BLOCK statement or a SURFACE statement.

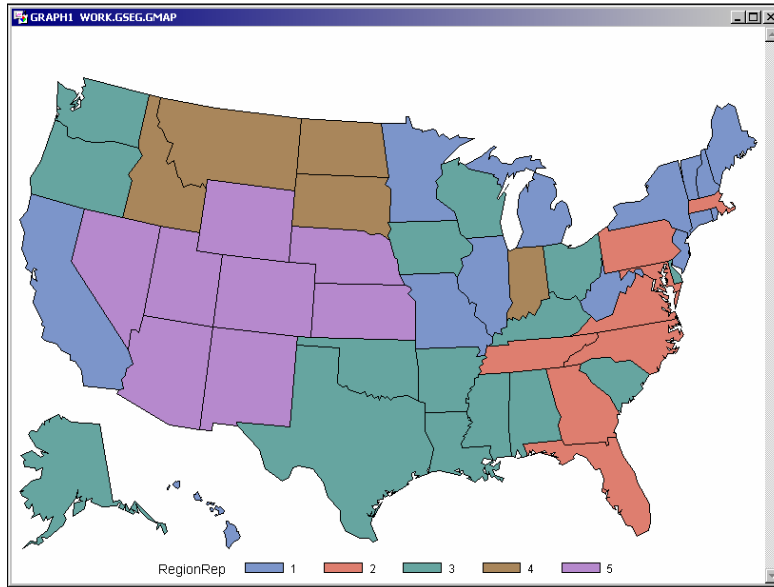
This paper will focus on creating the Choropleth map. The geographic map data for the US is found in **MAPS.US**. The sales rep data is found in **ORION.SITES**. Since a choropleth map is desired, the CHORO statement will be used. Areas will be shaded different colors depending on the value of **RegionRep**, which is the response variable.

```

proc gmap map=maps.us data=orion.sites;
  id state;
  choro regionRep;
run;
quit;

```

When the code is submitted, the output is not sent to the output window. A new window will open, called the GRAPH window, which contains the output graph.

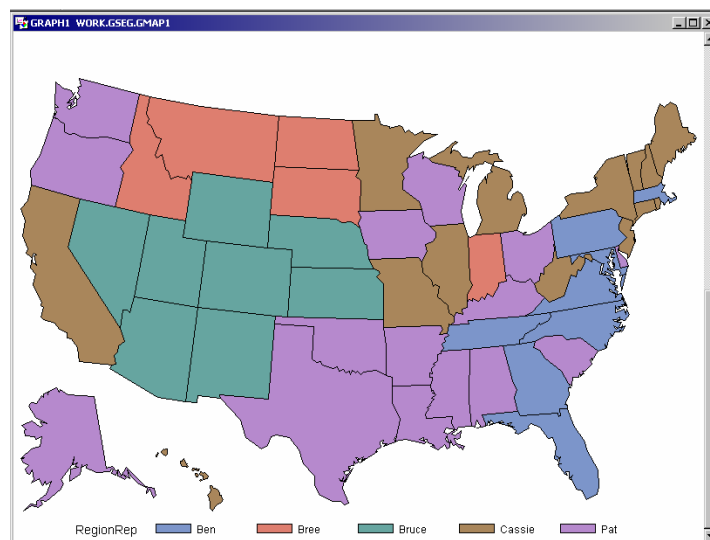


Since it is difficult to recall who is assigned what **RegionRep** value, it would be nice to have the person's name displayed. This can be done by adding a **FORMAT** statement using the user defined format **\$territory**.

```

proc gmap map=maps.us data=orion.sites;
  id state;
  choro regionRep;
  format regionRep $territory.;
run;
quit;

```

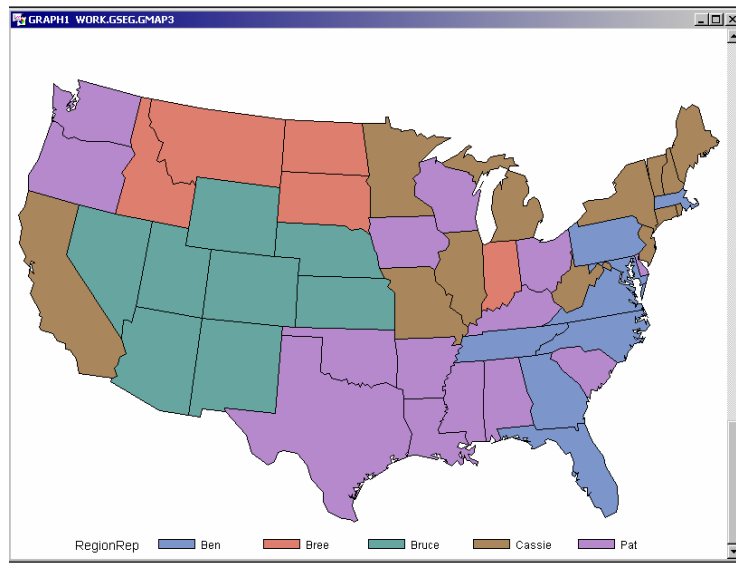


ELIMINATING STATES

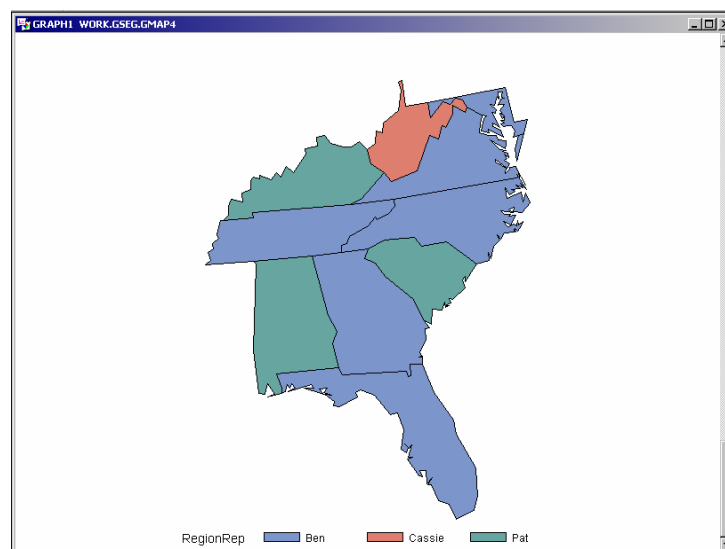
Perhaps we only want to display a portion of the United States. You can refine what regions are displayed easily using the WHERE statement. The WHERE statement will allow you to subset or filter your data. This is the same statement that is used in other procedures, such as Proc Print, Proc Report and Proc Means, to name a few.

For example, perhaps Alaska and Hawaii should not appear in the graph.

```
proc gmap map=maps.us data=orion.sites;  
  id state;  
  choro regionRep;  
  format regionRep $territory.;  
  where stcode not in ('AK','HI');  
run;  
quit;
```



Perhaps only states in the Southeastern United States should be displayed. Consider the states in the SESUG region:



This can be created by modifying the WHERE statement in the code above to the following:

```
where stcode in ('FL','GA','SC','NC','VA','TN','AL','KY','MD','WV');
```

CONTROLLING THE COLORS USED IN THE MAPS

It is a bit difficult to distinguish Pat's territory from Ben's because the colors are very similar. This can easily be rectified using graphics options known as GOPTIONS. These graphics options control attributes of the graph- such as colors, fonts, patterns, and size. They can also affect the settings of device parameters such as the type of output produced and the destination of the output.

The GOPTIONS statement is a global SAS statement. This means that options set on this statement remain in effect until you change them or cancel them. Please note the GOPTIONS statement only affects graphic options, and should not be confused with the OPTIONS statement. Note also that the GOPTIONS procedure will list the current settings of all graphics options.

First, we must choose the colors we want to use. Cassie will appear as pink, Ben as light blue, Pat as purple, Bree as orange, and Bruce as green.

To do so, the GOPTIONS statement will be added to the code with the COLORS= option. The order of the colors is important, so that they are assigned to the correct person. (You can find more about the colors choices available in the SAS/GRAPH portion of SAS OnlineDoc.)

The syntax of the GOPTIONS statement using the COLORS= option is:

```
goptions colors=( <colors desired separated by a space> );
```

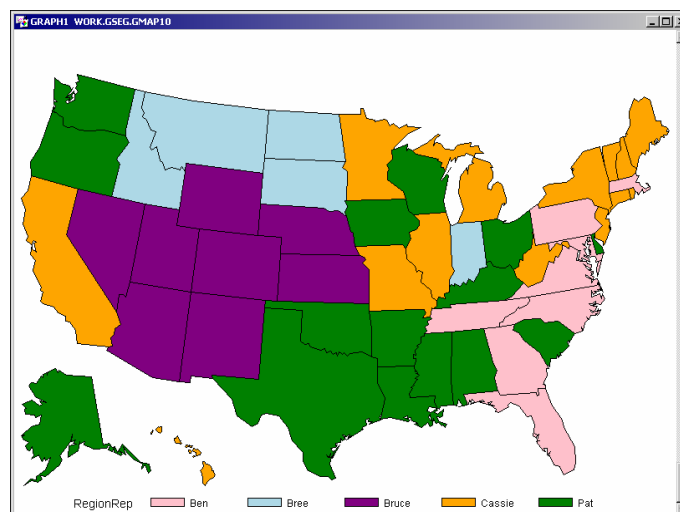
Recall the format:

```
proc format;  
  value $territory 1= 'Cassie'  
                  2= 'Ben'  
                  3= 'Pat'  
                  4= 'Bree'  
                  5= 'Bruce';  
  
run;
```

Taking the order into account, the following code is written and submitted:

```
goptions colors=(pink lightblue purple orange green);  
proc gmap map=maps.us data=orion.sites;  
  id state;  
  choro regionRep;  
  format regionRep $territory.;  
run;  
quit;
```

This code produces the following graph:



Why is Ben pink? Clearly, Cassie was to be assigned the pink color! Is it because of the order of the data? Examining the data shows this is not the case. The first row has a **RegionRep** value of 3, which is Pat's value. However, Pat's territories are not appearing as pink.

VIEWTABLE: Orion.Sites			
	stcode	RegionRep	state
1	AK	3	2
2	AL	3	1
3	AR	3	5
4	AZ	5	4
5	CA	1	6
6	CO	5	8
7	CT	1	9
8	DE	3	10
9	FL	2	12
10	GA	2	13
11	HI	1	15
12	IA	3	19
13	ID	4	16

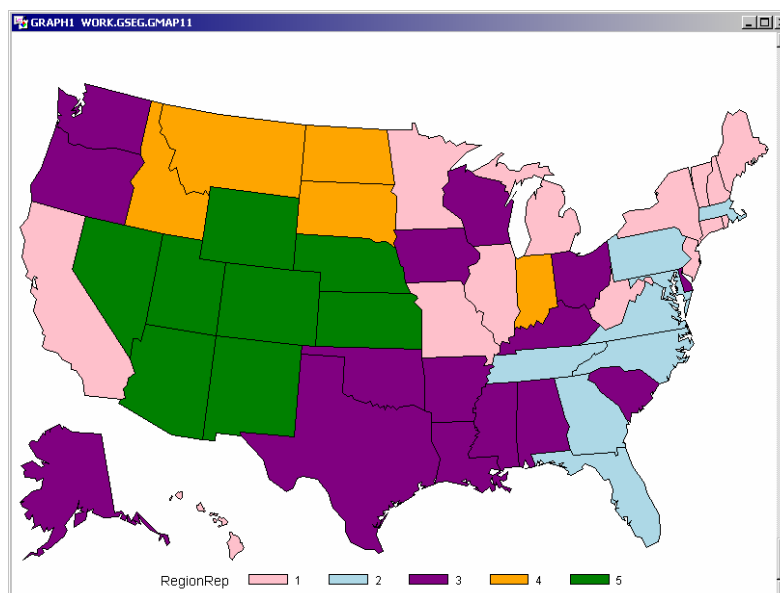
So why did the colors get assigned this way? For some reason, Ben was assigned the first color, Bree the second, Bruce the third, and so on. Do you notice a trend? It seems that colors were assigned in alphabetical order. Why?

There is a statement that affects the order that was not considered, and that statement changes what the output of that variable looks like... the **FORMAT** statement. If the statement is removed, notice how the colors change.

```

goptions colors=(pink lightblue purple orange green);
proc gmap map=maps.us data=orion.sites;
  id state;
  choro RegionRep;
run;
quit;

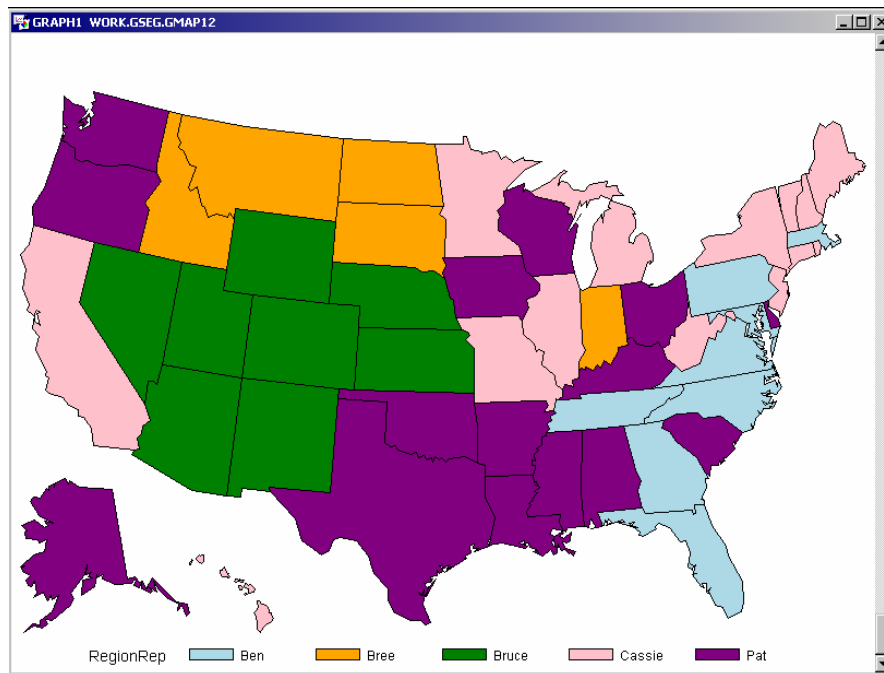
```



Notice in the legend that 1 is now pink, 2 is lightblue, 3 is purple, and so on. So the question then becomes how do we get the order desired, and still use the format? The answer is to change the order of the colors on the

GOPTIONS statement so that it matches the alphabetical order of the names. Ben is first alphabetically, so lightblue should be listed first, followed by Bree's orange, Bruce's green, Cassie's pink and Pat's purple.

```
goptions colors=(lightblue orange green pink purple);  
proc gmap map=maps.us data=orion.sites;  
  id state;  
  choro regionRep ;  
  format regionRep $territory.;  
run;  
quit;
```



Notice the colors are now correctly assigned.

It is useful to note that if only four colors were assigned, SAS would use a default color for the fifth value (i.e., Pat). Use caution when assigning colors, as the default color SAS assigns to values whose colors are unspecified might well be a color you assigned!

CHANGING THE OUTLINE OF THE STATES

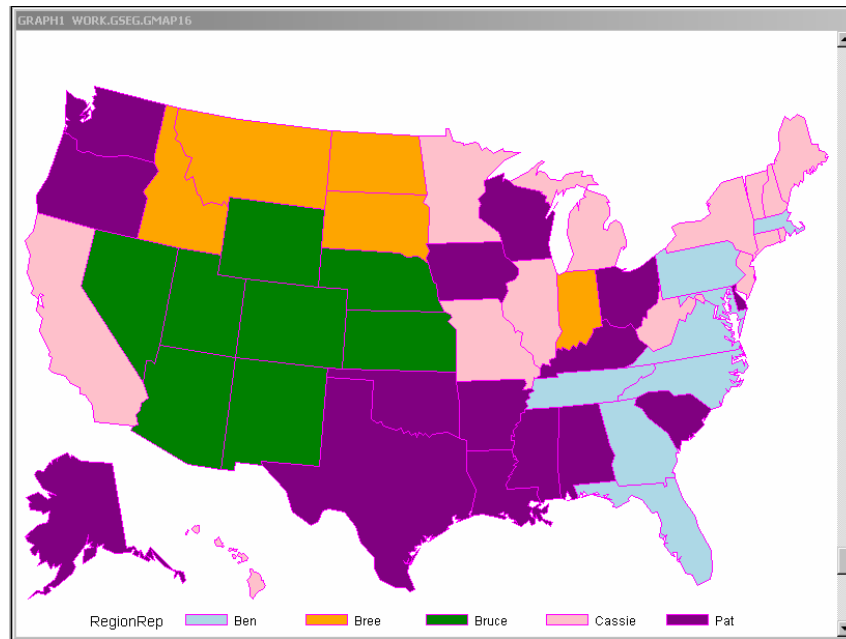
There are many ways to change the default output, and these can be explored more thoroughly in the SAS OnlineDoc.

The default color to outline the boundaries (in this example, the states) is black. It can easily be changed to another color as an option on the CHORO statement. Options can be assigned on the statement after the response variable by adding a single forward slash, similar to the DEFINE statement in the REPORT procedure. The options must be listed after the forward slash. The COUTLINE= option will assign a color to be used in the outline.

The following code changes the color of the outline to magenta. It is useful to note that if SAS cannot find the color requested, it will use gray as the color, and write a note in the log.

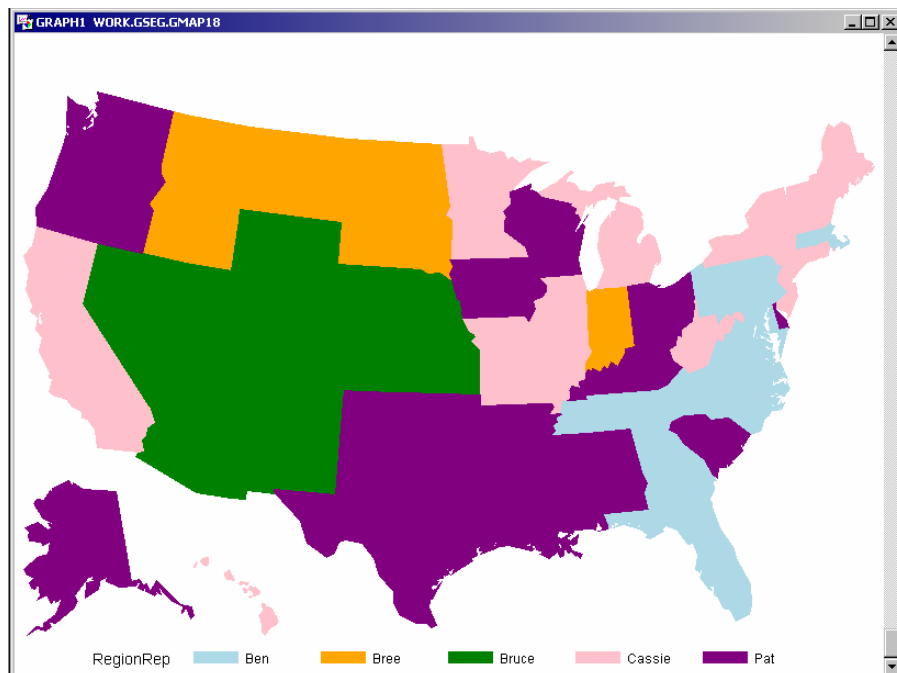
```
proc gmap map=maps.us data=orion.sites;  
  id state;  
  choro regionRep / coutline=magenta;  
  format regionRep $territory.;  
run;  
quit;
```

Since the graphics colors are globally set, we do not need to resubmit the GOPTIONS statement.



Perhaps no outline is desired at all. This can easily be removed by using the keyword **SAME** in place of the color after the COUTLINE= option.

```
proc gmap map=maps.us data=orion.sites;
  id state;
  choro regionRep / coutline=same;
  format regionRep $territory.;
run;
quit;
```



Notice that the lines around the states do not appear in the output. SAS is using the color of the state to draw the boundary lines.

RESETTING GRAPHIC OPTIONS

Graphics options can be reset using the RESET= option on the GOPTIONS statement. All options can be reset to their default values, or specific option(s) can be reset by using this option.

The syntax is:

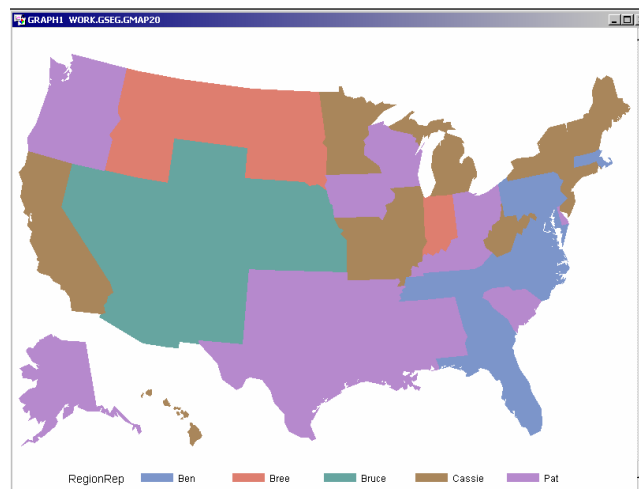
```
options reset= < ALL | GLOBAL | GOPTIONS > ;
```

The ALL option resets all the graphics options and global non-graphic options like titles and footnotes. The GLOBAL option does not reset the graphics options, but does reset all non-graphic statements like titles and footnotes. The GOPTIONS option resets the graphic options without resetting the non-graphic statements.

Example code:

```
options reset=goptions;  
proc gmap map=maps.us data=orion.sites;  
  id state;  
  choro regionRep / coutline=same;  
  format regionRep $territory.;  
run;  
quit;
```

Since the GOPTIONS were reset, the colors previously assigned have been set back to the default values.



In order for the graphic options to be reset for the next graphical output, this statement must appear **before** the Proc GMAP code. Also, it is possible to list more than one option on the GOPTIONS statement, similar to the OPTIONS statement. Please note that the RESET=ALL or RESET=GOPTIONS must be the first option specified in the GOPTIONS statement! If not, the graphics options that are listed before RESET= in the GOPTIONS statement are reset. Other options can follow the RESET= graphics option in the statement.

CREATING A GIF FILE OF YOUR MAP

It would be useful to save the map as a GIF file, or other graphic format, suitable for posting on a web page or SharePoint site, saving to a folder for later reference, including in an Information Delivery Portal or sending in an email.

This can be easily done in your SAS program by setting graphic device options on the GOPTIONS statement or by using ODS HTML.

METHOD 1: SETTING OPTIONS

The DEVICE= option allows you to specify the device type. In this example, a GIF file is desired. A destination file name is needed, and is specified on the GSFNAME= option. This must be a *fileref* (file reference), so a *fileref* is created on a FILENAME statement to specify the destination file.

The first two lines below were added to create the GIF file:

```
filename grafout1 'C:\Orion Star Maps\SalesRepTerritories.gif';
options device=gif gsfname=grafout1;

options colors=(lightblue orange green pink purple);
proc gmap map=maps.us data=orion.sites;
  id state;
  choro regionRep;
  format regionRep $territory.;
run;
```

METHOD 2: USE ODS HTML TO CREATE THE GIF

All that is needed is to wrap the GMAP procedure with two ODS HTML statements. The first statement opens the HTML destination. The BODY= option MUST be specified on this statement, and tells SAS the name of the file to create. The second ODS HTML statement should be placed after the QUIT statement to close the destination. If this is not done, any subsequent code that generates output will be appended to the end of the file.

```
ods html body='temp.html';
proc gmap map=maps.us data=orion.sites;
  id state;
  choro regionRep;
  format regionRep $territory.;
run;
quit;

ods html close;
```

SAS will create the file GMAPx.gif (where x is an integer starting with 1) in the current working directory, which is typically found in the bottom right hand corner of the display manager. It is helpful to set the location within the SAS program, especially when running in batch or command mode.

This can be done by adding an X statement **before** the ODS code is submitted.

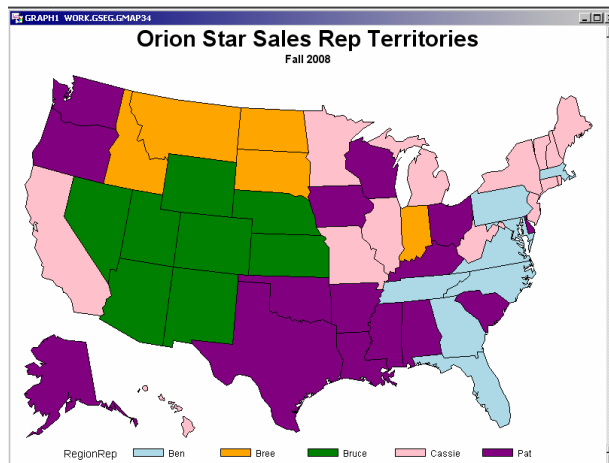
```
x 'cd C:\Orion Star Maps';
```

This statement executes an external command and is a great way to programmatically change the current working directory. This is not the only method, but the easiest to remember. SAS will then create the file in the Orion Star Maps folder on the C drive.

ADDING TEXT TO YOUR MAPS

The easiest texts to add to a map are titles and footnotes. These can be added by using TITLE and FOOTNOTE statements before the GMAP procedure. The statement is the same TITLE and FOOTNOTE statements that you may have used on other types of output, like listing and summary reports.

```
title1 'Orion Star Sales Rep Territories';
title2 'Fall 2008';
proc gmap map=maps.us data=orion.sites;
  id state;
  choro regionRep;
  format regionRep $territory.;
run;
quit;
```



Next, it would be helpful if state names appeared in the output. This can be done by using ANNOTATE= option on the CHORO statement.

A data set that contains the labels and their latitude and longitude locations is needed with the ANNOTATE= option. The ANNOTATE= option will overlay the labels on top of the map, so it is necessary that the data set be created before the GMAP procedure is submitted.

The following code was obtained from the SAS OnlineDoc for 9.1.3 in the “Labeling the States on a U.S. Map” example. It creates the data set **ORIONCENTER** which will be used in the GMAP procedure to produce the labels. Some of the two character abbreviations will appear in the state, but since some states are small, they will have lines drawn from the state to the abbreviation. These abbreviations will appear in the ocean area, which you will see mentioned in the code.

The new data set, **ORIONCENTER**, will be created from **MAPS.USCENTER** which provides coordinates for the labels. The **FIPSTATE** function converts the FIPS codes to the two character abbreviation of the state, which will be used as the labels. The **FLAG** variable, which is initially turned off, signals when labels should be written outside of the state. Also, because the variable called **WHEN** is “a” (denoting after), the labels are drawn after the map is drawn.

```
data orioncenter;
  length function $ 8;
  retain flag 0 xsys ysys '2' hsys '3' when 'a'
         style 'swiss';

  set maps.uscenter (where=(fipstate(state) ne 'DC') drop=long lat);

  function='label';
  text=fipstate(state);
  size=2.5;
  position='5';

  /* if the labeling coordinates are outside the state, OCEAN='Y'. The label is
     added and moves to get ready to draw the line from the state to the
     abbreviation. The FLAG variable is set to 1 so that the line will be
     drawn.*/
  if ocean='Y' then do;
    position='6';
    output;
    function='move';
    flag=1;
  end;

  /* If the label is to appear outside of the state, the line will be drawn
```

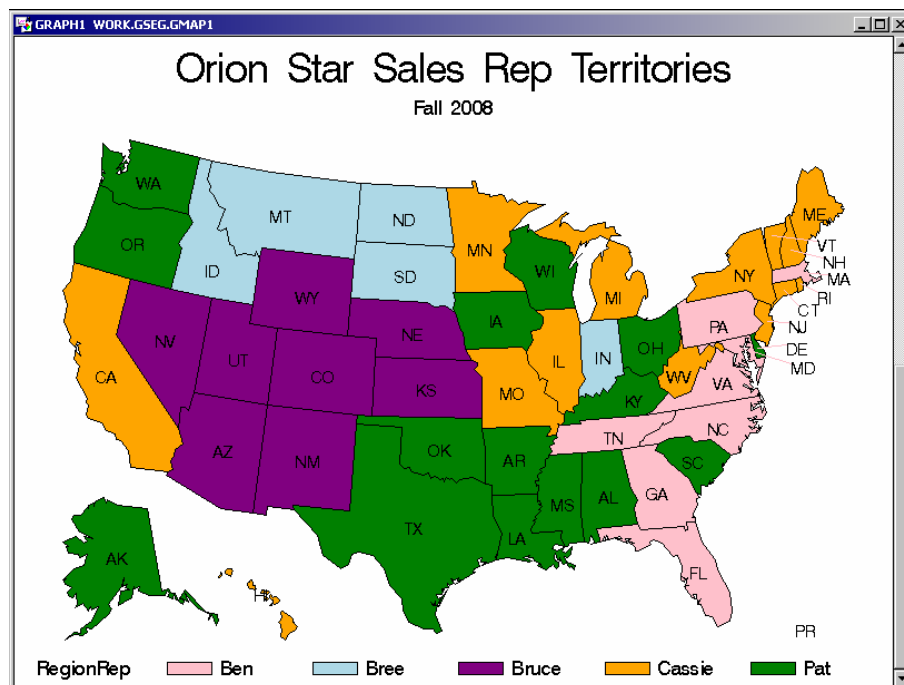
```

        and the flag to indicate the abbreviation is in the ocean is reset. */
    else if flag=1 then do;
        function='draw';
        size=.25;
        flag=0;
    end;
    output;
run;

```

Now that the data set is created, the ANNOTATE= option is added as an option on the CHORO statement. To ensure the text abbreviations are in black text, the CTEXT= option is set on the GOPTIONS statement.

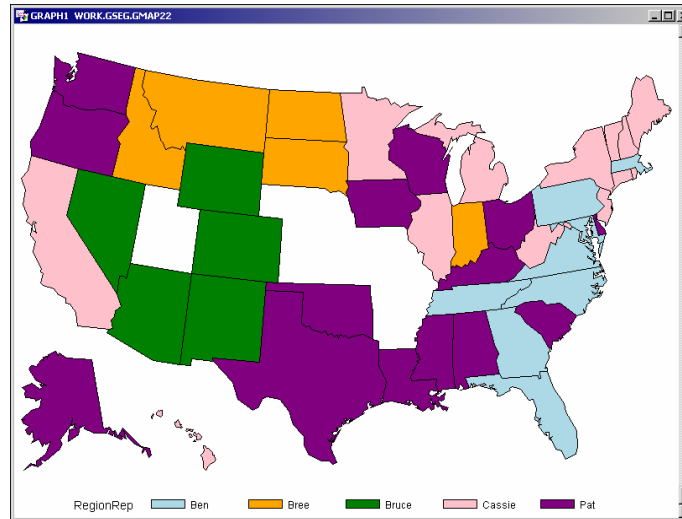
```
options ctext=black;
proc gmap map=maps.us data=orion.sites;
  id state;
  choro regionRep /annotate=orioncenter;
  format regionRep $territory.;
run;
quit;
```



WHEN THERE IS MISSING DATA

What would happen if some of the states were not present in the `ORION.SITES` data set? For example, what if Utah, Missouri, Arkansas, Nebraska and Kansas were not present in the response data set? By default, the GMAP procedure will not draw lines for these missing areas.

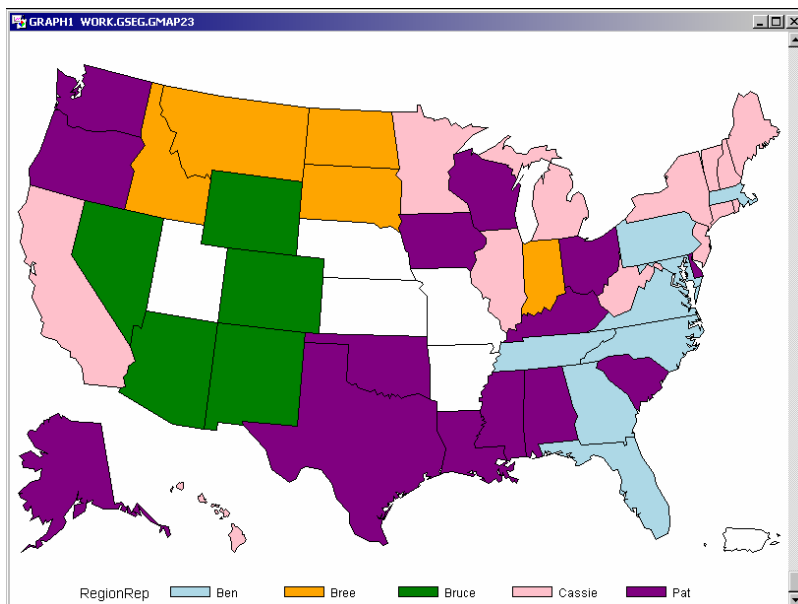
See below where the data set `ORION.PARTIAL` contains all the states in `ORION.SITES` except for aforementioned states.



It would be desirable to have the lines drawn for the missing data. One method would be to add the rows to the data set, but that would require knowing what was missing. This is not reliable, as human error could easily miss something- or perhaps the data is read only.

The easiest way is to let the GMAP procedure do the work for you using the ALL option. This option should appear on the PROC GMAP statement.

```
options colors=(lightblue orange green pink purple);
proc gmap map=maps.us data=orion.partial all;
  id state;
  choro regionRep;
  format regionRep $territory.;
run;
quit;
```



Notice Puerto Rico has now appeared in the map! It seems it was not present in the original data set.

OTHER LEVELS OF GEOGRAPHY

It might be nice to have a more detailed level on the graphical output- perhaps county or city information. This is also available in SAS/GRAPH.

For example, Ben has hired seven interns in Florida to help him; each intern is assigned several counties. Ben would like a map of Florida with each county represented. Each intern should get their own color and the map should only show Florida. Ben's data set contains an **InternCode** and the FIPS code for the county. There is one row for each county in the data set. The **InternCode** is similar to the **RegionRep** variable from **ORION.SITES**.

This is entirely possible using the methods described above. What will be different is that the **MAPS.COUNTIES** dataset will be used instead of the **MAPS.US**. The data set will first be subset to only include observations for Florida, which has a FIPS code of 12.

```
data florida;
  set maps.counties;
  where state=12;
run;
```

A format was created first, similar to the **\$territory** format .

```
proc format;
  value $interns '11'= 'Mango'
                 '12'= 'Wanda'
                 '13'= 'Tommy'
                 '14'= 'Bob'
                 '15'= 'DeShelia'
                 '16'= 'Qiwei'
                 '17'= 'Eli';
run;
```

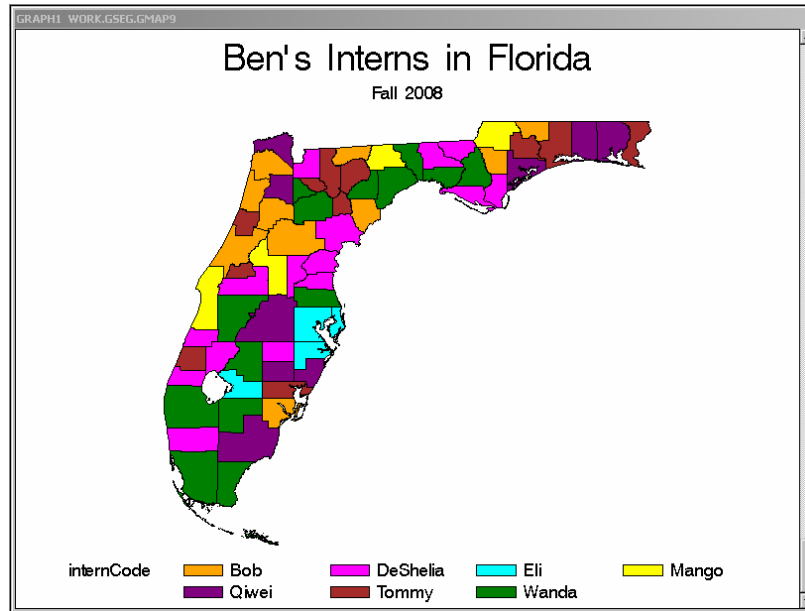
Colors will be assigned on a **GOPTIONS** statement and the **GMAP** procedure written with the **County** variable on the **ID** statement as the unit to graph. The response variable is the **InternCode**.

```
goptions colors=(orange magenta cyan yellow purple brown green);
title "Ben's Interns in Florida";
title2 'Fall 2008';

proc gmap map=florida data=orion.FLintern;
  id county;
  choro internCode;
  format internCode $interns.;
run;
quit;
```

Note: the data set did not have to be subset in a separate step. The alternative is to use a **WHERE=** option on the map data set on the Proc **GMAP** statement. The code is shown below. This is more efficient, as the map data is only passed through once. However, beginning users may find incremental code easier to follow, which is why it is included.

```
proc gmap map=maps.county(where=(state=12)) data=orion.FLintern;
  id county;
  choro internCode;
  format internCode $interns.;
run;
quit;
```



Notice that the map is backwards! This is easily fixed.

Some SAS-supplied map data sets are unprojected, and the **MAPS.COUNTY** data set is one of these. A data set that is unprojected is one where the x and y coordinates (longitude and latitude, respectively) of the boundaries are spherical coordinates. The desired map should display Cartesian coordinates, i.e., two dimensions. Maps using spherical coordinates are called unprojected, and maps using Cartesian coordinates are called projected.

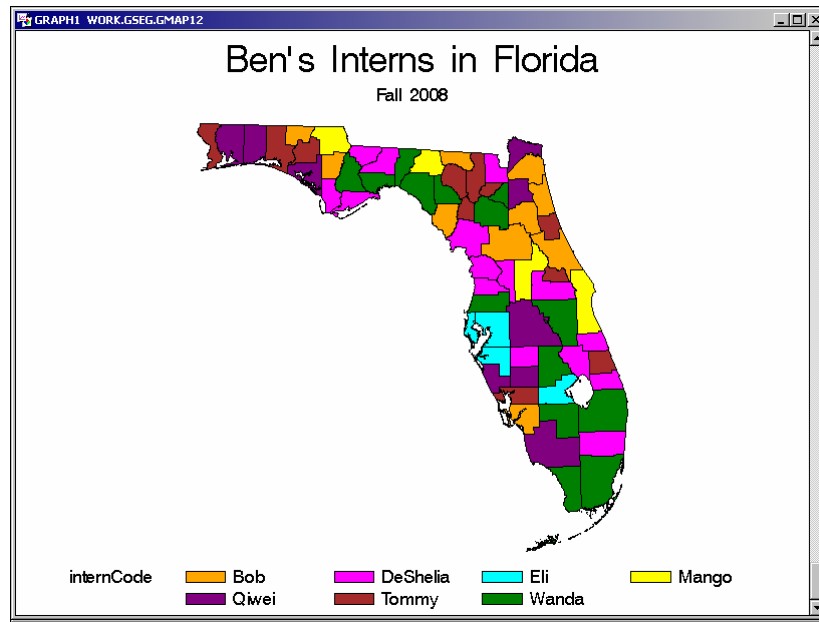
Basically, the rule of thumb is this: If your map is backwards, use the GPROJECT procedure.

This code should be placed above the GMAP procedure. The data set that contains the coordinates is the one that should be projected, in this example, the **FLORIDA** data set.

```
proc gproject data= <data set containing unprojected coordinates>
    out= <projected data set to be used on the MAP= option in PROC GMAP>;
    id < variable containing map unit>;
run;
```

The code is now modified to project the map data:

```
proc gproject data=florida out=florida;
    id county;
run;
proc gmap map=florida data=orion.FLintern;
    id county;
    choro internCode;
    format internCode $interns.;
run;
quit;
```



The map is fixed!

CONCLUSION

Visual display of graphical information is often easier to digest and communicate than simple listing and summary reports. This paper focused on Choropleth Maps and some of the graphic options available to you using SAS/GRAPH. More advanced maps can be produced using SAS/GIS, however, this software may not be available to you. This paper in no way covers everything, but attempts to introduce SAS users who have not worked with SAS/Graph maps to their syntax and concepts. Changes can be easily made to programs, making map generation quick and simple.

There are many great resources available to learn more: including previous SAS User Group Proceedings, SAS Online Documentation (jump straight to the SAS/GRAPH product) and the many great SAS Press offerings on the topic. SAS OnlineDoc also contains many examples with detailed explanation that you might find helpful.

REFERENCES AND RECOMMENDED READING

A good introductory book on producing graphs using SAS/GRAPH is:

Miron, Thomas, *The How-To Book for SAS/GRAPH Software*, SAS Press, June 1995.

There is also a SAS Maps Online Reference found at <http://support.sas.com/rnd/datavisualization/mapsonline/>.

This paper heavily relied on the SAS/GRAPH reference in SAS OnlineDoc (Version 9.1.3) which can be found at <http://support.sas.com/onlinedoc/913/docMainpage.jsp>. (Note that other versions of the documentation are also available online.)

The specific sections used are:

- SAS/Graph Concepts

- The Annotate Facility

- SAS/Graph Procedures

- The GMAP Procedure

- The GOPTIONS Procedure

- Example : Labeling the States on a U.S. Map

- <http://support.sas.com/onlinedoc/913/getDoc/en/graphref.hlp/gmplabel-ex.htm>

There are also examples found online in the SAS/GRAPH Samples Output Gallery at <http://support.sas.com/sassamples/graphgallery/index.html>.

More advanced graphing concepts are explored in more detail in a SUGI29 paper by Mike Zdeb entitled “*Creating Maps with SAS/GRAPH® - Drill Downs, Pop-Ups and Animation*” This paper was presented in the Hands-On Workshop Section, and is suitable for a user looking for more advanced options and uses. The paper can be found in the online proceedings for the conference at:

<http://www2.sas.com/proceedings/sugi29/120-29.pdf>

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name	Elizabeth Ceranowski
Enterprise	SAS Institute Inc.
Address	SAS Campus Drive
City, State ZIP	Cary, NC 27513
Work Phone:	(919)531-9347
E-mail:	Elizabeth.Ceranowski@sas.com
Web:	www.sas.com/student

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.