Delivering geographic information: For those who can't read a GMAP and won't stop to ask for directions

Jeanne Spicer
The Pennsylvania State University

Abstract

Many have lost their way on the road to spatial visualization. This tutorial plots a course through the steps necessary to map US data by state, county, and other levels. The route followed is to generate the basic map with SAS/ASSIST, getting as close to your final destination as possible. Then the finishing touches are added by modifying the code ASSIST generates. Common roadblocks are discussed including: hitting potholes in getting your response data ready to map and what to do when you get pulled over by the GDEVICE squad. The paper takes a detour through other PROCS that are often used with GMAP to demonstrate combining county boundaries with GREMOVE and customizing a map with ANNOTATE data sets. Also included are side-trips though the uncharted territory of using boundary files that are not part of the MAP LIBRARY and the new MAP object in SAS/AF. The paper concludes with a pit stop to discuss whether or not to tow the whole thing over to SAS/GIS.

Introduction

Documentation for PROC GMAP is intimidating to many new users. But producing maps can be easy – if you know where to start. The easiest way to start mapping is via SAS/ASSIST. Observe closely how ASSIST builds the code for PROC GMAP as you point & click your way through the windows. You are going to be working with two types of files:

Response Data

A file containing the variables to be displayed on a map is referred to as a "response data set". Any SAS data set or the output from a SAS statistical procedure can provide the variables to be represented on your map. Output from summary procedures like PROC Rank, Means or Summary is particularly suited to mapping. Residuals from multivariate regression analyses are fascinating when displayed on a map. The response variables can be character or numeric (continuous or categorical). The response data set must contain one or more ID variables that identify the geographic location where each value should be plotted.

Map Data

The SAS/Graph map library contains boundary files for the world and over 75 individual countries; these are referred to as "map data sets". Also included in the library are "idname data sets" that provide place names and their coordinates for labeling your maps. These files are described in SAS Technical Report P-196. When choosing a map data set from the manual you must make sure that it has been *projected*. A projected map is based on Cartesian coordinates and is ready to be displayed on a flat surface. An unprojected map data set contains the latitudes and longitudes of the area, but has not determined how the boundaries are to be laid out on paper. Before using them, you must run them through PROC GPROJECT. Unprojected map data is used when a particular projection method is required to reduce distortion or when putting several different map files together.

Map boundary files are very large and are not normally loaded when SAS is installed. The map files are compressed on the installation CD and must be uncompressed when they are loaded onto your system. Maps that come with SAS should all be stored in the MAP library, which on a PC is usually the c:\SAS\Maps directory. Like the SASUSER library, the LIBREF for this library should be assigned automatically when you log in.

Linking Map and Response Data

For SAS to display your data on a map, both your response data and map data must contain an identification variable (or variables) having the same name and type. SAS is *very* touchy about this and you will find yourself spending lots of time creating variables that are compatible. Two useful skills are the use of the RENAME statement and the ability to convert variables from character to numeric.

International boundaries are linked by a variable named COUNTRY that contains the World Geographic Code. If internal divisions are required, you will also need a variable named ID that contains a code for the appropriate sub-division.

US state level boundaries are identified by the variable, STATE that contains the 2-digit state FIPS code. If county boundaries are to be depicted, you will also need a second variable named COUNTY, containing the 3-digit county FIPS code. Several SAS functions are available to convert state names, abbreviations and zip codes to FIPS codes.

Many of the map data sets have a related "template" file. This file can be merged with your response data to ensure that your response data includes the appropriate ID variables.

Test Drive

Before you begin to map your own data, take the time to check and see if your system is set up correctly to map US data using a standard map data set that is provided with SAS/GRAPH: MAPS.US and a familiar data set from the SASUSER library: SASUSER.CRIME. This exercise will allow you to make sure that the map data sets are accessible, that you are able to navigate the GMAP windows in SAS/ASSIST, that the options for displaying and printing are set correctly, that you can edit and print your map, and that you can save the code generated by ASSIST.

1. Check to see if a MAPS libref is automatically assigned on your system by clicking the file cabinet icon when you start-up SAS. If you have no MAPS libref, you will have to issue a libname statement for the directory that contains your map files. Once the libref has been assigned, look in the library for the following map data sets: US & USCOUNTY. If no maps appear, it may be that they have not been loaded or uncompressed.
2. Now enter SAS/ASSIST and follow the selection sequence GRAPH→HIGH RESOLUTION→ MAPS. Try to create a plot of the incidence of auto theft in the lower 48 states.

   Active Data Set: SASUSER.CRIME
   Variable to Map: AUTO
   Area to Map: United States → Lower 48
   RUN

   After clicking each of the options, pause to view the code generated by ASSIST and learn how each button relates to the syntax of the GMAP procedure. *See Appendix B: PROC GMAP syntax locator.* Code can be viewed by pressing the RESULTS button and selecting View Current Program.
3. The GRAPHICS DEVICE button allows you to specify the display device used in the GRAPHICS window and optionally a hardcopy device. ASSIST is generating the DEVICE= and TARGETDEVICE= parameters for the GOPTIONS statement here. The window seems to be asking you the same thing twice. You are asked where to "Send Output To:" MONITOR or HARDCOPY and then there are 2 buttons which allow you to select the drivers to be used. Always start by sending the output to MONITOR or the output will go directly to the hardcopy device without giving you a chance to view it. If you don't specify the name of a hardcopy device, the map will be displayed with the colors that your monitor is capable of producing. If the map is to be printed, have SAS try to emulate how the map would look on a particular printer by keeping "Send Output to: MONITOR", but specifying the name of a hardcopy device by clicking the Hardcopy Device button. See if the color contrast is OK and look in your log window to find out if substitutions were made for the colors or fonts that you have requested. If you are using a networked printer and must print your map to a GACCESS file, you can insert the specifications for writing to this file by including GOPTIONS code in the Driver Management Supplemental Prefix commands window.
4. When the map appears, poke around in the GRAPH window. From the menu bar EDIT→ EDIT GRAPH, you can edit the map in a "paintbrush–like" environment. Be sure to print your map from the FILE menu in this window. Those who are using a networked printer will do the same, but the map will not actually be printed, it will be written to a file. Then use a system print command to print it.
5. Be sure to save the code for the map that ASSIST has generated. Go through the SAS/GRAPH manual and look up the syntax for the statements in this program. You won't be able to do all your mapping from ASSIST, but the code for these simple maps will be provide you with a base for your own programs.

Mapping Your Own Data

Once you are comfortable working with the sample data, try using your own response data. For example, an extract of data labor data contains the variables[1]:

```
Fips5   State   LMA    Menclus          Womclus
5.      $2.     5.     8.               8.
------------------------------------------------
1001    AL      281    1                6
1003    AL      282    1                6
1005    AL      288    3                4
1007    AL      284    2                2
1009    AL      284    2                2
1011    AL      281    1                6
etc
```

The response data is *not* ready to be mapped. The variable FIPS5 in this file contains the FIPS codes for both state and county in a single number. There is a

[1] Copies of the data & programs used in this paper are available via anonymous FTP at ftp.pop.psu.edu cd /pub/spicer/.

variable in the file named STATE, but it’s character and contains the postal abbreviation for the state name. So, before entering ASSIST to map the data we will make a pass through it with the following code:

```
0001    data usdata (drop=cstate);
0002    set stf.labor (rename=(state=cstate));
0003    length state county 5.;
0004    state= stfips(cstate);
0005    county = substr(put(fips5,z5.),3,3);
0006    run;
```

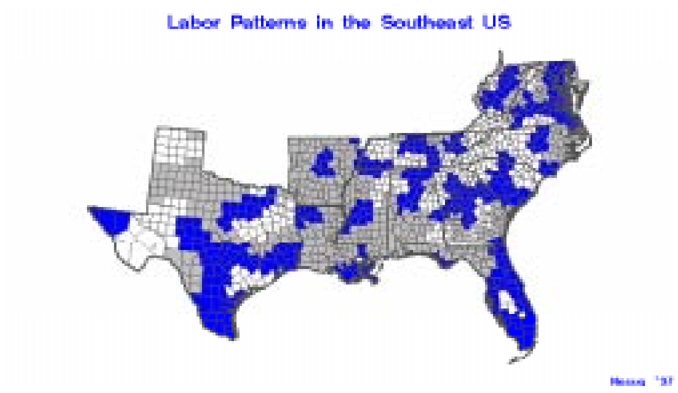
Line 2 reads the postal abbreviation in with a new name “CSTATE”. The ID variables required for US mapping are declared as numeric variables in line 3. The SAS function STFIPS in line 4 converts the postal abbreviation to a state FIPS code. We obtain the county FIPS from the last 3 digits of the variable FIPS5 in line 5. Note that FIPS5 has been output to the substring function using the z5. format. This ensures that the last 3 digits of a 4 or 5 digit FIPS code will be pulled off correctly for both Napa, CA fips5 = 6055 and Sagadahoc, ME fips5 = 23023.

Now that the data set contains the proper ID variables, we can go into the ASSIST map window.

Active Data Set: WORK.USDATA
Variable to Map: MENCLUS
Area to Map: United States
Additional Options: “2D/Prism Options”,
“Include County Boundaries”
[Note: This option will not appear unless you have selected a response data set that does indeed contain a numeric variable named COUNTY.]

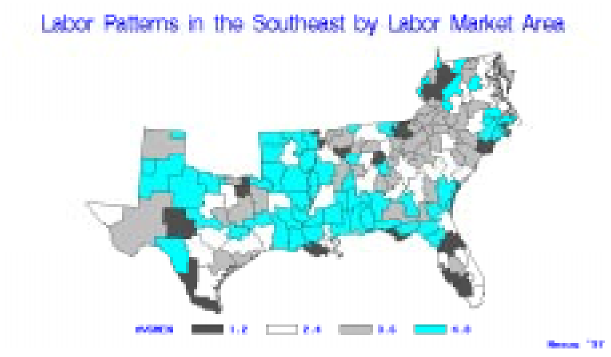
When you examine the code for this map, you will notice that by requesting County boundaries, SAS has chosen a different Map Data Set: USCOUNTY.

Detailing



County maps are great, but if your map displays more than one state you would like to be able to distinguish state from county borders. There is code to accomplish this in the SAS SAMPLE LIBRARY: *Annotated State Lines on County Map* which uses ANNOTATE data sets and unprojected map files to overlay more pronounced state borders on top of the county borders. It’s a complicated program and hopefully future versions of GMAP will include this function as a standard option. The program has no response data. In order to modify this code for your own data, you need to specify the states you want (or remove the subsetting IF’s for the entire country in both data steps. Then substitute the name of your response data set in the DATA= option of the PROC GMAP and supply the desired response variable in CHORO statement.

Custom Conversions



The data we have been using includes a third geographic identifier code: LMA. “Labor market areas” are groups of several contiguous counties. To map the data for each area, the values to be plotted must be aggregated so that the response data set contains only one observation for each LMA. To depict the boundary of an LMA you want to use the outside boundaries of the counties that are in the LMA, but remove the internal boundaries using PROC GREMOVE. You will need to attach the LMA values to each observation in the USCOUNTY map dataset before running GREMOVE, so that SAS knows which counties to combine. *See Appendix A.*

Off-Road

If the area you need to map is not included in the standard SAS library, you may be able to find the appropriate boundary files in a public access archive on the web, for example, CIESIN at
ftp.ciesin.org
cd /pub/census/usa/tiger
This archive has many boundary files and the SAS code that will turn them into valid data sets. If you cannot locate a boundary file for your area, but have a printed map, you can have the map digitized and use the file with GMAP. Consult SAS Technical Report A-107.

Touring

Map Objects
In version 6.12, maps can be created within graphical objects in an AF screen. Try it with a state level map of Auto Theft. To display this information in a Map Object,

go into the build window for a SAS/AF frame entry. Select Actions → Make → Map object. A tabbed window will appear. Click the *Map* tab and supply the SASUSER.US map data set. Select the "Type of Map: Area". Click the *Response* tab and supply the name SASUSERS.CRIME. Select AUTO as the "Color variable". Click the *Legend* tab and request "Legend". Click "OK" and view the Map Object in your frame. SAS has a default algorithm for displaying the data, but you can create a RANGE entry in the AF catalog to define the range of values to be displayed by each color.

GIS

Of course the hot topic these days is the new SAS/GIS product. Users with an academic license are lucky, it's included in the package, but other users will have to consider the additional cost vs. the benefits over SAS/GRAPH mapping. SAS/GIS basically provides you with more depth in the information you can display on a map. A GIS map stores map and response data in different layers that are linked together by the geographic ids. The layers can be displayed or hidden dynamically using SAS/GIS. If you are unsure, contact SAS and arrange for a trial license. Start SAS/GIS from the pull-down menus by selecting: GLOBALS → PRESENT → SAS/GIS and go through the tutorial.

You might also like to try and bring in some sample TIGER data files that are free from the Bureau of Census on the web at location:
http://www.census.gov/geo/www/tiger/t195sets.html

You will get a zip file that will explode into 16 files. Be sure that you are using the COMPRESS=NO option in your Display Manager session. From the File menu, select Import. The "Import Type" is TIGER. In the field "Basic Data Record File Name" type the path and filename of the first file: TGR*nnnnn*.BW1. SAS will import this file and locate the other related files it needs in that directory. The import process takes quite a long time. When the process has finished, click "Return" to view the map. Click to display layers for railroads, streets, water sources and other features. You'll either say "Gotta have it" or "I can do what I need with GMAP".

References:

SAS Institute Inc., *SAS/GRAPH Software: Reference, Volumes 1 & 2*, SAS Institute, Cary, NC: 1990.

SAS Institute Inc., *SAS Technical Report P-196 SAS/GRAPH Software: Map Data Sets*, Release 6.06.

SAS Institute Inc., *SAS Technical Report A-107 Creating Your Own SAS/GRAPH Map Data Sets with a Digitizer*, SAS Institute, Cary, NC: 1982.

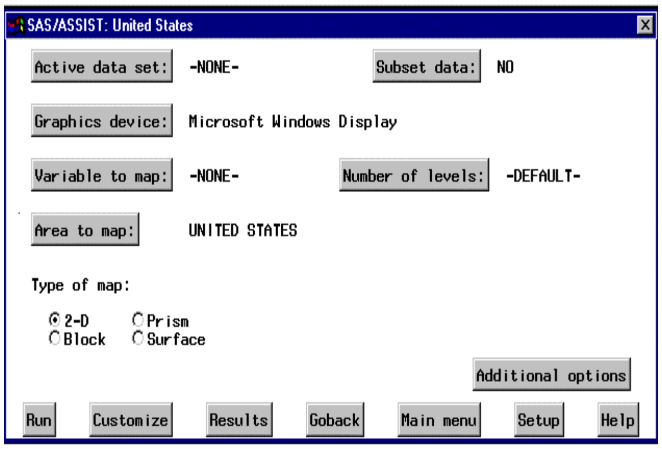
SAS Institute Inc., *Doing More with SAS/ASSIST, Version 6, First Edition*, SAS Institute, Cary, NC: 1992.

Appendix A – Aggregating county borders

```
* convert id variables;
 data lmas (keep=lma county state menclus);
 set myhome.labor (rename=(state=cstate));
 length state county 5.;
 state=stfips(cstate);
 county=substr(put(fips,z5.),3,3);
 run;

* attach lma values to maps.uscounty file;
proc sql;
create table joinlma as
select * from maps.uscounty m, lmas r
where m.state = r.state and m.county = r.county;
quit;

•  Take out county boundaries, leave lma;
proc sort data=joinlma out=lmasort; by lma; run;
proc gremove data=lmasort
 (where=(state in(  1 5 11 12 13 22 24 28 37 45
47 48 51 54 ))) out=rmvctys;
by lma;
id state county; run;

* Aggregate response variables;
proc summary data=lmas;
class lma;
var menclus;
output out=stats mean=avgmen; run;

*specify print options;
goptions  norotate hpos=0 vpos=0 device=WIN
ctext=blue graphrc interpol=join;

title1 "Labor Patterns in the Southeast by Labor
Market Area";
footnote1   j=RIGHT "Nesug '97";

pattern1 color=CHARCOAL value=MSOLID;
pattern2 color=WHITE value=MSOLID;
pattern3 color=GRAY value=MSOLID;
pattern4 color=cyan value=MSOLID;

* response dataset from proc summary and map
dataset from proc gremove;
proc gmap data=stats (where=(_type_= 1))
          map=rmvctys
          all;
   id lma;
   choro avgmen/
   coutline=DAGRAY
   levels=4;
run; quit;
```

Appendix B

PROC GMAP Syntax Locator


SAS/ASSIST: United States

Active data set: -NONE- Subset data: NO

Graphics device: Microsoft Windows Display

Variable to map: -NONE- Number of levels: -DEFAULT-

Area to map: UNITED STATES

Type of map:

[✓ 2-D] Prism
Block Surface

PROC GMAP
MAP= *mapdataset*;
DATA = *responsedataset* ;
	Area to map:
	Active data set:

CHORO *variables* ; SURFACE *variables* ; BLOCK *variables*; PRISM *variables* ;	Type of map: ◉ 2-D ○ Prism ○ Block ○ Surface Variable to map:
ID *variables*;	*Determined by →* Area to map: *and* Additional options 2D/PRISM OPTIONS *(for US counties)*
LEGEND*n* <*options*>; PATTERN*n* <*options*>;	Additional options *Legend options* *Pattern options*
TITLE*n* <*options*> <'*text*'>; FOOTNOTE*n* <*options*> '<*text*'>;	Customize
Options:	

DISCRETE
LEVELS= *n*
MIDPOINTS= *values*
	Number of levels:

COUTLINE= *color*
XSIZE= *n* PCT|n CM|*n* IN
YSIZE= *n* PCT|n CM|*n* IN
MISSING
CTEXT= *color*
CEMPTY= *color*
	Additional options *2D/ Prism options*

Printing:	
GOPTIONS TARGETDEVICE= DISPLAY=	Graphics device: