

CREATING MAPS WITH SAS

TASC/Advanced Support Team
Center for Information Technology
National Institutes of Health

December 18, 2000

Index

- Creating Maps With SAS/GGRAPH
 - Map Data Sets
 - Response Data Sets
 - Template Data Sets
 - Other Data Sets
 - Documentation
- Mapping Procedures
 - GMAP
 - GPROJECT
 - GREDUCE
 - GREMOVE
- PATTERN Statement
- Annotating Maps
- Sample Maps
 - Outline of the United States
 - Mapping a Character Response Variable
 - Mapping a Numeric Response Variable
 - Redefining Regions
 - Thick Lines Outline Regions
 - Redefining Regions with Regions Separated
 - Maryland and Virginia Counties
 - All States with Some Counties
 - Projecting and Reducing a Map Data Set
 - United States Cities
 - Labels Inside the Map
 - Labels Outside the Map
- Available Maps
- Contents of Map Data Sets
- FIPS Codes
- World Geographic Codes
- "Producing Graphs with SAS" Course Notes

Creating Maps with SAS/GRAFH

The SAS/GRAFH module of the SAS System provides several [procedures](#) that allow you to draw maps. You can draw just the [outline of the map](#) or you can [display your data on it](#). In this short course you can learn the basics of drawing maps with SAS and view some sample maps and programs by choosing [Sample Maps](#) above. First, you should become familiar with some terms used when drawing maps.

Map Data Sets

Data is provided with SAS/GRAFH to draw the maps of most countries. This data is saved in the form of SAS data sets. They are called **map data sets**. On the PC, they are saved in the MAPS directory that is located under the SAS directory. On the NIH MVS mainframe, they are stored in the SAS library ZABCRUN.SAS810.MAPS. [More on map data sets](#).

Response Data Sets

The data that you represent on your map is called a **response data set**. This SAS data set must contain at least one identification variable that is also contained in the map data set. The identification variables are used to match your data to the regions of the map. [More on response data sets](#).

Template Data Sets

For most countries there is another data set that contains the names of the regions within a country and an identification number associated with it. These data sets are called **template data sets**. They are usually used for labelling a map. The names of these data sets usually start with the first 7 characters of the country's name followed by the number 2. For example, the template data sets for Argentina and China are called ARGENTI2 and CHINA2. [More on template data sets](#).

Other Data Sets

In addition to map and template data sets, SAS/GRAFH provides the following data sets:

ANOMALY	contains known changes to the US states and counties
USCENTER	contains the coordinates for the visual center of US states. See example .
USCITY	contains the coordinates of some US cities. See example .

Documentation

The full description of the procedures and statements used here appear in "SAS/GRAFH Software: Reference", Version 8. This guide is available in our website <http://statsoft.nih.gov>. To learn more about SAS/GRAFH you can attend the CIT course "Producing Graphs with SAS", or view the class notes in the Training section of this website.

Map Data Sets

Many map data sets are included with SAS/GGRAPH. PC users can choose to install all or some of these data sets. If you don't install all of them you can always install them later.

Map data sets contain the coordinates necessary to draw maps. The coordinates can be **projected** or **unprojected**. Unprojected coordinates contain latitude and longitude in radians. Projected coordinates contain Cartesian coordinates. Coordinates are stored in the variables X and Y. Map data sets have to be projected before you use them to create a map. Use the procedure GPROJECT to do this. Almost all map data sets are projected but a few are not (e.g. COUNTIES).

Most map data sets have been **reduced**. They maintain the general appearance of the map but contain fewer points than the originally digitized map. A few map data sets are not reduced (e.g. COUNTIES and STATES). To reduce a map data set you can use the procedure GREDUCE and/or the variable DENSITY.

To draw a map you use the procedure GMAP. To assign colors or patterns to the regions use PATTERN statements. If you want to annotate your map you can use SAS's Annotate Facility or you can create a graphics file that you can edit with another software.

The table in Available Maps contains a list of all available map data sets in SAS 8.0. It also indicates if a map data set is projected and/or reduced, and what is its template data set. To view a list of the variables for any of these data sets select Contents of Map Data Sets.

Response Data Sets

The data that you represent on your map is called a response data set. This SAS data set must contain at least one identification variable that is also contained in the map data set. The identification variables are used to match your data to the regions of the map. They must be specified in the ID statement of PROC GMAP.

Specify the name of your response data set after the DATA= option in PROC GMAP. The name of the response variable is indicated in the CHORO statement.

To assign colors or patterns to the regions use PATTERN statements.

The program below reads the variable VOTE from the response data set VOTES. The ID is STATE and exists in both data sets MAPS.US and in VOTES. The variable STATE must contain the state's fips codes.

```
data votes;
  input stcode $ vote $ pop;
  state=stfips(stcode);
  cards;
CA Yes 123
ID Yes 240
UT Yes 350
TX Yes 100
NE Yes 250
MN Yes 370
IL No 380
WI No 340
GA No 160
VA No 255
;

pattern1 v=msolid c=red;
pattern2 v=msolid c=blue;

proc gmap map=maps.us data=votes all;
  id state;
  choro vote / coutline=black;
run;
```

[See example of mapping a character response variable.](#)

[See example of mapping a numeric response variable.](#)

Template Data Sets

Template data sets are SAS data sets provided with SAS/GRAFTH that contain the names and the identification numbers of the unit areas of a map. This information is usually stored in the variables **IDNAME** and **ID**, respectively. You can view the variable names in the section [Contents of Map Data Sets](#).

A list of all map data sets and their corresponding template data sets appears in [Available Maps](#).

Most template data sets for the countries of North America, Central America and South America also contain the variables listed below. These variables are used for labelling the maps using the Annotate facility of SAS/GRAFTH. An [example](#) is available in the section [Sample Maps](#).

Variable	Description
MID_X	x-coordinate of geographic midpoint
MID_Y	y-coordinate of geographic midpoint
OUT_X	x-coordinate for outside labelling
OUT_Y	y-coordinate for outside labelling

With the United States maps you can use the [USCITY](#) and the [USCENTER](#) data sets to include city names and state names in the correct locations. View the contents of these data sets in [Contents of Map Data Sets](#). There are two examples that illustrate the use of these data sets in [Sample Maps](#).

Mapping Procedures

Procedure Name	Description	Examples
GMAP	Produces two-dimensional or three-dimensional maps	<u>1-12</u>
GPROJECT	Converts spherical coordinates into Cartesian coordinates	<u>9</u>
GREDUCE	Used to reduce the number of coordinates used to draw a map	<u>9</u>
GREMOVE	Used to combine unit areas into larger unit areas	<u>4, 5, 6, 8</u>

Other: PATTERN Statement

GMAP Procedure

(**MVS users** need to include the following LIBNAME statement to allocate the map data sets:
libname maps 'zabcrun.sas810.maps' disp=shr;)

The GMAP procedure produces two-dimensional and three-dimensional maps. The general syntax for two-dimensional maps is shown below. The GMAP procedure is fully documented in "SAS/GGRAPH Software: Reference", Version 8.

For help online, enter **help gmap** in the command box of the SAS System for Windows.

```
proc gmap map=maps.mapds data=respds options;
  id idvars;
  choro respvars / options;
run;
```

Here,

- mapds* is the name of the map data set you want to use,
- respds* is the name of the response data set that contains your data,
- options lists additional option(s) you request,
- idvars* indicates the variable(s) that identify the unit areas of the map, and,
- respvars* specifies the response variable you want to represent on the map.

Examples: [1-12](#)

Options Used in Sample Maps

PROC GMAP Statement Options

- ALL** Specifies that all unit areas from the map data set be drawn even if there is no matching data in the response data set

CHORO Statement Options

- ANNO=** points to the annotate data set that will be used for labelling (see the Annotate Facility in "SAS/GGRAPH Software: Reference", Version 8)
- COUTLINE=** specifies the color to use to outline each unit area
- DISCRETE** specify this option if your response variable is numeric and discrete
- LEGEND=** points to the LEGEND statement that should be used to modify the default legend (see the course notes for ["Producing Graphs with SAS"](#) in the Training section of this website).
- NOLEGEND** indicates that no legend should be produced

PATTERN Statement

The **PATTERN** statement is used with GMAP to assign colors and/or patterns to the unit regions of the map.

PATTERN Statement

The PATTERN statement defines patterns and colors for the unit areas of the map. It is of the form:

```
PATTERNn COLOR=color VALUE=pattern REPEAT=m ;
```

where n can be any number from 1 to 99. If n is not specified, 1 is assumed.

The COLOR= option specifies a valid color for the device. It may be abbreviated as C=. If neither the COLOR= option nor the CPATTERN graphics option are used, the PATTERN statement cycles through each color in the device's colors list before the next PATTERN statement is used.

The VALUE= option specifies the pattern to use for the unit areas. It can be abbreviated as V=. The valid values for maps are:

MEMPTY	Requests an empty pattern (abbreviated as E)
MSOLID	Requests a solid pattern (abbreviated as S)
Mtsa	Draws parallel or crosshatched lines: t=thickness, t=1,2,3,4,5 (1 is the lightest) s=style, s=N (parallel lines), X (crosshatched lines) a=angle, a=0 to 360 (angle at which lines are drawn from horizontal) (e.g. M1N0, M2X45, M5X90)

The REPEAT= option specifies the number of times a PATTERN is applied before using the next PATTERN statement.

PATTERN statements are assigned to the values of the response variable in alphabetical or numerical order of the values.

PATTERN statements stay in effect until you reset them with a GOPTIONS statement or until you specify new patterns or cancel them with a PATTERN statement with no options.

Example

GPROJECT Procedure

(**MVS users** need to include the following LIBNAME statement to allocate the map data sets:
libname maps 'zabcrun.sas810.maps' disp=shr;)

The GPROJECT procedure converts spherical coordinates (radians) into Cartesian coordinates. Map data sets have to be projected before you use them to create a map.

The general syntax of the procedure is shown below. The complete syntax is documented in "SAS/GRAFH Software: Reference", Version 8.

For help online, enter **help gproject** in the command box of the SAS System for Windows.

```
proc gproject data=maps.mapds out=outds options;
  id idvars;
run;
```

Here,

- mapds* is the name of the map data set you want to project,
- outds* is the new map data set created by GPROJECT,
- options* lists additional option(s) you request, and,
- idvars* indicates the variable(s) that identify the unit areas of the map.

Example

GREDUCE Procedure

(**MVS users** need to include the following LIBNAME statement to allocate the map data sets:
libname maps 'zabcrun.sas810.maps' disp=shr;)

The GREDUCE procedure is used to reduce the number of points used to draw a map. It creates a map data set with the variable DENSITY. This variable can be used to subset the data. It has values from 0 to 6.

The general syntax of the procedure is shown below. The complete syntax is documented in "SAS/GRAFH Software: Reference", Version 8.

For help online, enter **help greduce** in the command box of the SAS System for Windows.

```
proc greduce data=maps.mapds out=outds options;  
  id idvars;  
run;
```

Here,

- mapds* is the name of the map data set you want to reduce,
- outds* is the new map data set created by GREDUCE,
- options* lists additional option(s) you request, and,
- idvars* indicates the variable(s) that identify the unit areas of the map.

Note: Some map data sets already contain the variable DENSITY.

Example

GREMOVE Procedure

(**MVS users** need to include the following LIBNAME statement to allocate the map data sets:
libname maps 'zabcrun.sas810.maps' disp=shr;)

The GREMOVE procedure is used to combine unit areas into larger unit areas.

The general syntax of the procedure is shown below. The complete syntax is documented in "SAS/GRAFH Software: Reference", Version 8.

For help online, enter **help greduce** in the command box of the SAS System for Windows.

```
proc gremove data=maps.mapds out=outds;
  id idvars;
  by byvars;
run;
```

Here,

mapds is the name of the map data set to use,

outds is the new map data set created by GREMOVE,

idvars indicates the variable(s) that identify the current unit areas of the map, and,

byvars indicates the variable(s) that identify the new unit areas.

Examples: [4](#), [5](#), [6](#), [8](#)

Annotating Maps

You can draw symbols and labels on your map in various ways:

- with the SAS Annotate Facility, or,
- by creating a CGM, GIF or JPEG file then editing it with an editor (e.g. WordPerfect, Word).

SAS Annotate Facility

The Annotate Facility is used to enhance graphs created by SAS/GRAFH. You create a data set of graphics commands that it uses to annotate the graph. To learn more about the Annotate Facility read Chapters 10 and 11 of the "SAS/GRAFH Software: Reference", Version 8.

The following examples use the Annotate Facility: [5](#), [9](#), [10](#), [11](#) and [12](#).

Creating CGM, GIF and JPEG Files

You can annotate a graph by first creating a CGM, GIF or JPEG file containing your graph. Then you could annotate it using another software such as Microsoft Word, WordPerfect or some other editor that reads GIF and JPEG files.

To learn how to create these files see the course notes for "[Using SAS to Publish Static Web Pages](#)" or the course notes for "[Producing Graphs with SAS](#)".

SAS/GRAFH Map Data Sets

SAS Version 8

Map Data Set	Country	Projected/ Reduced	Template Data Set
AFGHANIS	Afghanistan	Y/Y	AFGHANI2
AFRICA	Africa	Y/Y	NAMES
ALGERIA	Algeria	Y/Y	ALGERIA2
ANDORRA	Andorra	Y/Y	ANDORRA2
ARGENTIN	Argentina	Y/Y	ARGENTI2
ARMENIA	Armenia	Y/Y	ARMENIA2
ASIA	Asia	Y/Y	NAMES
AUSTRAL	Australia	Y/Y	AUSTRAL2
AUSTRIA	Austria	Y/Y	AUSTRIA2
AZERBAIJ	Azerbaijan	Y/Y	AZERBAI2
BANGLADE	Bangladesh	Y/Y	BANGLAD2
BELARUS	Belarus	Y/Y	BELARUS2
BELGIUM	Belgium	Y/Y	BELGIUM2
BELIZE	Belize	Y/Y	BELIZE2
BHUTAN	Bhutan	Y/Y	BHUTAN2
BOLIVIA	Bolivia	Y/Y	BOLIVIA2
BOSNIAHE	Bosnia/Herzegovina	Y/Y	BOSNIAH2
BRAZIL	Brazil	Y/Y	BRAZIL2
BRUNEI	Brunei	Y/Y	BRUNEI2
BULGARIA	Bulgaria	Y/Y	BULGARI2
CAMBODIA	Cambodia	Y/Y	CAMBODIA2
CANADA	Canada (provinces and census districts)	Y/Y	CANCENS
CANADA2	Canada (provinces)	Y/Y	NONE
CANADA3	Canada (provinces and census districts)	N/N	CANCENS
CANADA4	Canada (provinces)	N/N	NONE
CHILE	Chile	Y/Y	CHILE2
CHINA	China, Hong Kong, Taiwan	Y/Y	CHINA2
COLOMBIA	Colombia	Y/Y	COLOMBI2
COSTRICA	Costa Rica	Y/Y	COSTRIC2
COUNTIES	United States (counties) and Puerto Rico (towns)	N/N	CNTYNAME
COUNTY	United States (counties)	N/Y	NA
CROATIA	Croatia	Y/Y	CROATIA
CUBA	Cuba	Y/Y	CUBA2
CYPRUS	Cyprus	Y/Y	CYPRUS2
CZECH	Czechoslovakia	Y/Y	CZECH2
DENMARK	Denmark	Y/Y	DENMARK2
DOMINREP	Dominican Republic	Y/Y	DOMINRE2
ECUADOR	Ecuador	Y/Y	ECUADOR2

EGYPT	Egypt	Y/Y	EGYPT2
ETHIOPIA	Ethiopia	Y/Y	ETHIOP12
EUROPE	Europe	Y/Y	NAMES
FGUIANA	French Guiana	Y/Y	FGUIANA2
FINLAND	Finland	Y/Y	FINLAND2
FRANCE	France, Monaco	Y/Y	FRANCE2
GAZASTRI	Gaza Strip	Y/Y	GAZASTR2
GEORGIA	Georgia	Y/Y	GEORGIA2
GERMANY	Germany	Y/Y	GERMANY2
GREECE	Greece	Y/Y	GREECE2
GUATEMAL	Guatemala	Y/Y	GUATEMA2
GUYANA	Guyana	Y/Y	GUYANA2
HAITI	Haiti	Y/Y	HAITI2
HKDB1	Hong Kong	Y/Y	HKDB2
HKKWDB1	Hong Kong	Y/Y	HKKWDB2
HKKWP1	Hong Kong	Y/Y	HKKWP12
HKNTDB1	Hong Kong	Y/Y	HKNTDB2
HKNTPU1	Hong Kong	Y/Y	HKNTPU2
HKPU1	Hong Kong	Y/Y	HKPU2
HONDURAS	Honduras	Y/Y	HONDURA2
HUNGARY	Hungary	Y/Y	HUNGARY2
INDIA	India, Pakistan, Bhutan, Nepal, Sri Lanka, Bangladesh	Y/Y	INDIA2
INDONESI	Indonesia	Y/Y	INDONES2
IRAN	Iran	Y/Y	IRAN2
IRAQ	Iraq	Y/Y	IRAQ2
IRELAND	Ireland	Y/Y	IRELAND2
ISRAEL	Israel	Y/Y	ISRAEL2
ITALY	Italy, San Marino	Y/Y	ITALY2
IVORY	Ivory Coast	Y/Y	IVORY2
JAMAICA	Jamaica	Y/Y	JAMAICA2
JAOSAKA	Osaka, Japan	Y/Y	JAOSAKA2
JAPAN	Japan	Y/Y	JAPAN2
JATOKYO	Tokyo, Japan	Y/Y	JATOKYO2
KAZAKHST	Kazakhstan	Y/Y	KAZAKHS2
KENYA	Kenya	Y/Y	KENYA2
KOREANOR	Korea, North	Y/Y	KOREANO2
KOREASOU	Korea, South	Y/Y	KOREASO2
KOSEOUL	Seoul, Korea	Y/Y	KOSEOUL2
KYRGYZST	Kyrgyzstan	Y/Y	KYRGYZS2
LAOS	Laos	Y/Y	LAOS2
LATVIA	Latvia	Y/Y	LATVIA2
LEBANON	Lebanon	Y/Y	LEBANON2
LIBYA	Libya	Y/Y	LIBYA2
LIECHTEN	Liechtenstein	Y/Y	LIECHTE2
LITHUANI	Lithuania	Y/Y	LITHUAN2

LUXEMBOU	Luxembourg	Y/Y	LUXEMBO2
MACAU	Macau	Y/Y	MACAU2
MACEDONI	Macedonia	Y/Y	MACEDON2
MALAYSIA	Malaysia, Singapore, Brunei	Y/Y	MALAYSI2
MEXICO	Mexico	Y/Y	MEXICO2
MOLDOVA	Moldova	Y/Y	MOLDOVA2
MONACO	Monac	Y/Y	MONACO2
MONGOLIA	Mongolia	Y/Y	MONGOLI2
MOROCCO	Morocco	Y/Y	MOROCCO2
MYANMAR	Myanmar	Y/Y	MYANMAR2
NAMERICA	North America	Y/Y	NAMES
NEPAL	Nepal	Y/Y	NEPAL2
NICARAGU	Nicaragua	Y/Y	NICARAG2
NIGERIA	Nigeria	Y/Y	NIGERIA2
NORWAY	Norway	Y/Y	NORWAY2
NIGERIA	Nigeria	Y/Y	NIGERIA2
NZ	New Zealand	Y/Y	NZ2
PAKISTAN	Pakistan	Y/Y	PAKISTA2
PANAMA	Panama	Y/Y	PANAMA2
PARAGUAY	Paraguay	Y/Y	PARAGUA2
PERU	Peru	Y/Y	PERU2
PHILIPPI	Philippines	Y/Y	PHILIPP2
POLAND	Poland	Y/Y	POLAND2
PORTUGAL	Portugal	Y/Y	PORTUGA2
RUSSIA	Russia	Y/Y	RUSSIA2
SALVADOR	El Salvador	Y/Y	SALVADO2
SAMERICA	South America	Y/Y	NAMES
SANMARIN	San Marino	Y/Y	SANMARI2
SAUDIARA	Saudi Arabia	Y/Y	SAUDIAR2
SINGAPOR	Singapore	Y/Y	SINGAPO2
SLOVAKIA	Slovakia	Y/Y	SLOVAKI2
SLOVENIA	Slovenia	Y/Y	SLOVENI2
SPACIFIC	South Pacific	Y/Y	NAMES
SPAIN	Spain	Y/Y	SPAIN2
SRILANKA	Sri Lanka	Y/Y	SRILANK2
STATES	United States and Puerto Rico	N/N	NONE
SURINAME	Suriname	Y/Y	SURINAM2
SWEDEN	Sweden	Y/Y	SWEDEN2
SWITZERL	Switzerland	Y/Y	SWITZER2
SYRIA	Syria	Y/Y	SYRIA2
TAJIKIST	Tajikistan	Y/Y	TAJIKIS2
THAILAND	Thailand	Y/Y	THAILAN2
TAIWAN	Taiwan	Y/Y	TAIWAN2
TRINIDAD	Trinidad and Tobago	Y/Y	TRINIDA2
TUNISIA	Tunisia	Y/Y	TUNISIA2
TURKEY	Cyprus, Lebanon, Siria, Turkey	Y/Y	TURKEY2

TURKMENI	Turkmenistan	Y/Y	TURKMEN2
UK	United Kingdom	Y/Y	UK2
UKRAINE	Ukraine	Y/Y	UKRAINE2
URUGUAY	Uruguay	Y/Y	URUGUAY2
US	United States (states)	Y/Y	NONE
USCOUNTY	United States (counties)	Y/Y	CNTYNAME
UZBEKIST	Uzbekistan	Y/Y	UZBEKIS2
VENEZUEL	Venezuela	Y/Y	VENEZUE2
VIETNAM	Vietnam	Y/Y	VIETNAM2
WESTBANK	West Bank	Y/Y	WESTBAN2
WORLD	World	Y/Y	NA
YUGOSLAV	Yugoslavia	Y/Y	YUGOSLA2
ZAIRE	Zaire	Y/Y	ZAIRE2

Other Data Sets Used with Maps

Data Set Name	Description
ANOMALY	known changes of US states and counties
USCENTER	coordinates of visual centers of states used for labelling
USCITY	coordinates of some US cities used for labelling

(partial listing)

The CONTENTS Procedure

Data Set Name:	MAPS.AFGHANI 2	Observations:	29
Member Type:	DATA	Variables:	4
Engine:	V8	Indexes:	0
Created:	14: 58 Tuesday, August 10, 1999	Observation Length:	74
Last Modified:	14: 58 Tuesday, August 10, 1999	Deleted Observations:	0
Protection:		Compressed:	NO
Data Set Type:	SFT	Sorted:	NO
Label:	AFGHANISTAN 1998 SAS Institute Inc.	- Copyright(C)	

-----Engine/Host Dependent Information-----

Data Set Page Size:	8192
Number of Data Set Pages:	1
First Data Page:	1
Max Obs per Page:	110
Obs in First Data Page:	29
Number of Data Set Repairs:	0
File Name:	d:\SASV8\maps\afghani 2.sas7bdat
Release Created:	8.0000M0
Host Created:	WIN_NT

-----Alphabetic List of Variables and Attributes-----

#	Variable	Type	Len	Pos	Format	Label
2	COUNTRY	Num	5	64		World Geographic Code
3	ID	Num	5	69		Id Number
4	IDNAME	Char	35	29		Name of Internal Division
1	_MAP_GEOMETRY_	Char	29	0	SGEOREF.	Variable used by the Map Chart Client

The CONTENTS Procedure

Data Set Name:	MAPS.AFGHANIS	Observations:	2758
Member Type:	DATA	Variables:	6
Engine:	V8	Indexes:	0
Created:	14: 58 Tuesday, August 10, 1999	Observation Length:	34
Last Modified:	14: 58 Tuesday, August 10, 1999	Deleted Observations:	0
Protection:		Compressed:	NO
Data Set Type:		Sorted:	NO
Label:	Copyright(C) 1996 SAS Institute Inc.		

----- Engine/Host Dependent Information -----

Data Set Page Size: 4096
Number of Data Set Pages: 24
First Data Page: 1
Max Obs per Page: 119
Obs in First Data Page: 68
Number of Data Set Repairs: 0
File Name: d:\SASV8\maps\afghani.s.sas7bdat
Release Created: 8.0000M0
Host Created: WIN_NT

----- Alphabetic List of Variables and Attributes -----

#	Variable	Type	Len	Pos	Label
1	ID	Num	5	0	Id Number
6	LAT	Num	6	28	Deprojected latitude
5	LONG	Num	6	22	Deprojected longitude
2	SEGMENT	Num	5	5	Political Subdivision Segment Number
3	X	Num	6	10	Projected Longitude
4	Y	Num	6	16	Projected Latitude

US FIPS Codes

([FIPS functions](#))

FIPS Code	State
1	Alabama
2	Alaska
4	Arizona
5	Arkansas
6	California
8	Colorado
9	Connecticut
10	Delaware
11	District of Columbia
12	Florida
13	Georgia
15	Hawaii
16	Idaho
17	Illinois
18	Indiana
19	Iowa
20	Kansas
21	Kentucky
22	Louisiana
23	Maine
24	Maryland
25	Massachusetts
26	Michigan
27	Minnesota
28	Mississippi
29	Missouri
30	Montana
31	Nebraska
32	Nevada
33	New Hampshire
34	New Jersey
35	New Mexico
36	New York
37	North Carolina
38	North Dakota
39	Ohio
40	Oklahoma
41	Oregon

42	Pennsylvania
44	Rhode Island
45	South Carolina
46	South Dakota
47	Tennessee
48	Texas
49	Utah
50	Vermont
51	Virginia
53	Washington
54	West Virginia
55	Wisconsin
56	Wyoming
72	Puerto Rico

SAS FIPS and Postal Codes Functions

Function & Argument	Description
FIPNAME(fips)	converts FIPS code to state name (all uppercase)
FIPNAMEL(fips)	converts FIPS code to state name (in upper and lowercase)
FIPSTATE(fips)	converts FIPS state codes to two-character postal code
STFIPS(postalcode)	converts state postal codes to FIPS state codes
STNAME(postalcode)	converts state postal codes to state names (all uppercase)
STNAMEL(postalcode)	converts state postal codes to state name (upper and lowercase)
ZIPFIPS(zipcode)	converts ZIP codes to FIPS state codes
ZIPNAME(zipcode)	converts ZIP codes to state names (all uppercase)
ZIPNAMEL(zipcode)	converts ZIP codes to state name (upper and lowercase)
ZIPSTATE(zipcode)	converts ZIP codes to two-letter state codes

World Geographic Codes - Version 8

Code	Country	Code	Country
110	AFGHANISTAN	284	COCOS ISLANDS
120	ALBANIA	285	COLOMBIA
125	ALGERIA	286	COMOROS
60	AMERICAN SAMOA	290	CONGO
140	ANDORRA	293	COOK ISLANDS
141	ANGOLA	294	CORAL SEA ISLANDS
142	ANGUILLA	295	COSTA RICA
149	ANTI GUA/BARBUDA	440	CROATIA
150	ARGENTINA	300	CUBA
135	ARMENIA	305	CYPRUS
100	ARUBA	310	CZECH REPUBLIC
155	ASHMORE/CARTIER	315	DENMARK
160	AUSTRALIA	317	DJIBOUTI
165	AUSTRIA	318	DOMINICA
115	AZERBAIJAN	320	DOMINICAN REPUBLIC
180	BAHAMAS	325	ECUADOR
181	BAHRAIN	922	EGYPT
64	BAKER ISLAND	330	EL SALVADOR
182	BANGLADESH	332	EQUATORIAL GUINEA
184	BARBADOS	327	ERITREA
187	BASSAS DA INDIA	331	ESTONIA
211	BELARUS	335	ETHIOPIA
190	BELGIUM	334	EUROPA ISLAND
227	BELIZE	337	FALKLAND ISLANDS
311	BENIN	336	FAROE ISLANDS
195	BERMUDA	338	FIJI
200	BHUTAN	340	FINLAND
205	BOLIVIA	350	FRANCE
185	BOSNIA/HERZEGOVINA	355	FRENCH GUIANA
210	BOTSWANA	367	FRENCH POLYNESIA
212	BOUVET ISLAND	369	FRENCH SOUTHERN TERR.
220	BRAZIL	388	GABON
228	BRITISH INDIAN OCEAN	389	GAMBIA
231	BRITISH VIRGIN ISLANDS	393	GAZA STRIP
232	BRUNEI	390	GEORGIA
245	BULGARIA	394	GERMANY
927	BURKINA FASO	396	GHANA
252	BURUNDI	397	GIBRALTAR
255	CAMBODIA	399	GLORIOSO ISLANDS
257	CAMEROON	400	GREECE
260	CANADA	405	GREENLAND
264	CAPE VERDE	406	GRENADE
268	CAYMAN ISLANDS	407	GUADELOUPE
269	CENTRAL AFRICAN REP.	66	GUAM
273	CHAD	415	GUATEMALA
275	CHILE	416	GUERNSEY
280	CHINA	417	GUINEA
516	CHRISTMAS ISLAND	737	GUINEA-BISSAU
282	CLIPPERTON ISLAND	418	GUYANA

Code	Country	Code	Country
fffff	fffff	fffff	fffff
420	HAITI	73	MARSHALL ISLANDS
424	HEARD/MCDONALD	591	MARTINIQUE
430	HONDURAS	592	MAURITANIA
435	HONG KONG	593	MAURITIUS
65	HOWLAND ISLAND	594	MAYOTTE
445	HUNGARY	595	MEXICO
450	ICELAND	63	MICRONESIA
455	INDIA	71	MIDWAY ISLAND
458	INDONESIA	576	MOLDOVA
460	IRAN	607	MONACO
465	IRAQ	608	MONGOLIA
470	IRELAND	609	MONTSERRAT
475	ISRAEL	610	MOROCCO
480	ITALY	615	MOZAMBIQUE
485	IVORY COAST	250	MYANMAR
487	JAMAICA	821	NAMIBIA
488	JAN MAYEN ISLANDS	621	NAURU
490	JAPAN	61	NAVASSA ISLAND
62	JARVIS ISLAND	625	NEPAL
495	JERSEY	630	NETHERLANDS
67	JOHNSTON ATOLL	640	NETHERLANDS ANTILLES
500	JORDAN	645	NEW CALEDONIA
497	JUAN DE NOVA ISLAND	660	NEW ZEALAND
525	KAZAKHSTAN	665	NICARAGUA
505	KENYA	667	NIGER
68	KINGMAN REEF	670	NIGERIA
398	KIRIBATI	672	NIUE
514	KOREA, NORTH	683	NORFOLK ISLAND
515	KOREA, SOUTH	69	NORTHERN MARIANA ISLANDS
520	KUWAIT	685	NORWAY
510	KYRGYZSTAN	616	OMAN
530	LAOS	700	PAKISTAN
541	LATVIA	75	PALAU
540	LEBANON	70	PALMYRA ATOLL
543	LESOTHO	710	PANAMA
545	LIBERIA	712	PAPUA NEW GUINEA
550	LIBYA	714	PARACEL ISLANDS
553	LIECHTENSTEIN	715	PARAGUAY
542	LITHUANIA	720	PERU
570	LUXEMBOURG	725	PHILIPPINES
573	MACAU	727	PITCAIRN ISLANDS
574	MACEDONIA	730	POLAND
575	MADAGASCAR	735	PORTUGAL
577	MALAWI	72	PUERTO RICO
580	MALAYSIA	747	QATAR
583	MALDIVES	750	REUNION
585	MALI	755	ROMANIA
590	MALTA	825	RUSSIA
588	MAN	758	RWANDA

Code	Country	Code	Country
763	SAINT KITTS/NEVIS	651	VANUATU
775	SAINT VINCENT/GRENADINES	940	VENEZUELA
963	SAMOA	945	VIETNAM
782	SAN MARINO	78	VIRGIN ISLANDS (U. S.)
783	SAO TOME/PRINCIPE	80	WAKE ISLAND
785	SAUDI ARABIA	950	WALLIS/FUTUNA ISLANDS
787	SENEGAL	955	WEST BANK
788	SEYCHELLES	831	WESTERN SAHARA
790	SIERRA LEONE	965	YEMEN
795	SINGAPORE	970	YUGOSLAVIA
548	SLOVAKIA	291	ZAI RE
789	SLOVENIA	990	ZAMBIA
229	SOLOMON ISLANDS	818	ZIMBABWE
800	SOMALIA		
801	SOUTH AFRICA		
830	SPAIN		
833	SPRATLY ISLANDS		
272	SRI LANKA		
765	ST. HELENA		
770	ST. LUCIA		
773	ST. PIERRE/MIQUELON		
835	SUDAN		
840	SURINAME		
845	SVALBARD		
847	SWAZI LAND		
850	SWEDEN		
855	SWITZERLAND		
858	SYRIA		
281	TAIWAN		
784	TAJIKISTAN		
865	TANZANIA		
875	THAI LAND		
883	TOGO		
884	TOKELAU		
886	TONGA		
887	TRINIDAD AND TOBAGO		
889	TROMELIN ISLAND		
890	TUNISIA		
905	TURKEY		
909	TURKMENISTAN		
906	TURKS/CAICOS ISLANDS		
908	TUVALU		
910	UGANDA		
928	UKRAINE		
888	UNITED ARAB EMIRATES		
925	UNITED KINGDOM		
926	UNITED STATES		
930	URUGUAY		
931	UZBEKISTAN		

Sample Maps

Click on map to view program.

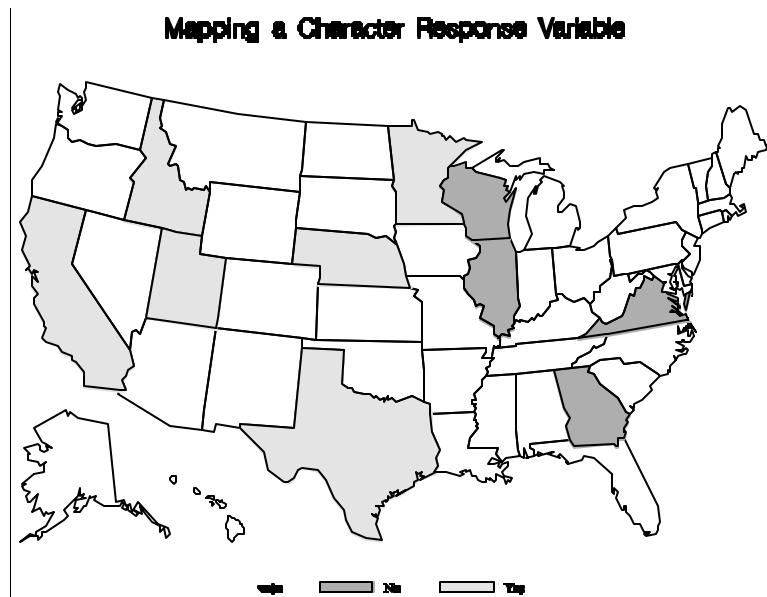
Outline of US



Outline of US with Hawaii and Alaska

```
* Outline of US with Hawaii and Alaska;  
  
* MVS users should remove the asterisk in the LIBNAME statement;  
* libname maps 'zabcrun.sas810.maps' disp=shr;  
  
goptions reset=all ftext=centb ftitle=swiss border;  
  
pattern v=mempty r=100;  
  
title 'Outline of US';  
  
proc gmap map=maps.us data=maps.us;  
  id state;  
  choro state / discrete nolegend coutline=black;  
run;  
quit;
```

Click on map to view program.



Mapping Character Data

```
* Mapping your data;
* Character response variable;

* MVS users should remove the asterisk in the LIBNAME statement;
* libname maps 'zabcrun.sas810.maps' disp=shr;

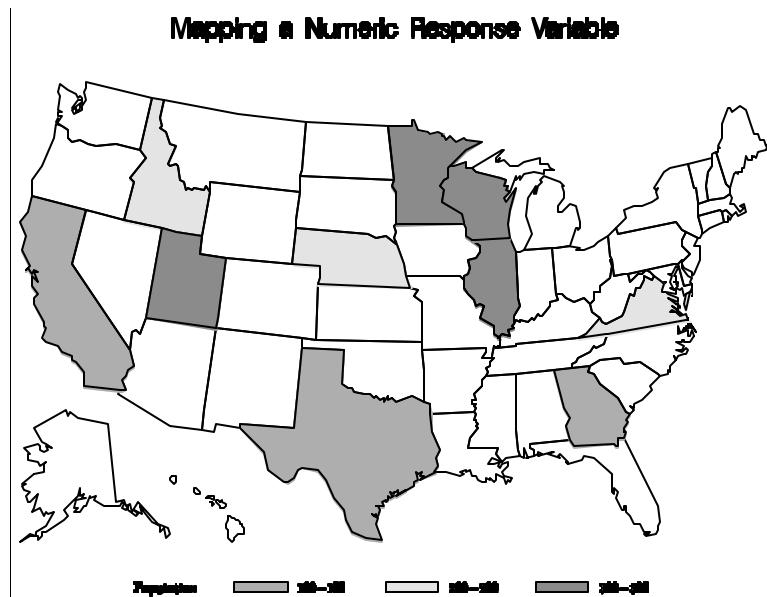
goptions reset=all ftext=centb ftitle=swiss border;

data votes;
  input stcode $ vote $ pop;
  state=stfips(stcode);
  cards;
CA Yes 123
ID Yes 240
UT Yes 350
TX Yes 100
NE Yes 250
MN Yes 370
IL No 380
WI No 340
GA No 160
VA No 255
VA No 130
;
pattern1 v=msolid c=liol;
pattern2 v=msolid c=bro;

title 'Mapping a Character Response Variable';

proc gmap map=maps.us data=votes all;
  id state;
  choro vote / coutline=black;
run;
quit;
```

Click on map to view program.



Mapping Numeric Data

```
* Mapping your data;
* Numeric response variable;

* MVS users should remove the asterisk in the LIBNAME statement;
* libname maps 'zabcrun.sas810.maps' disp=shr;

goptions reset=all ftext=centb ftitle=swiss border;

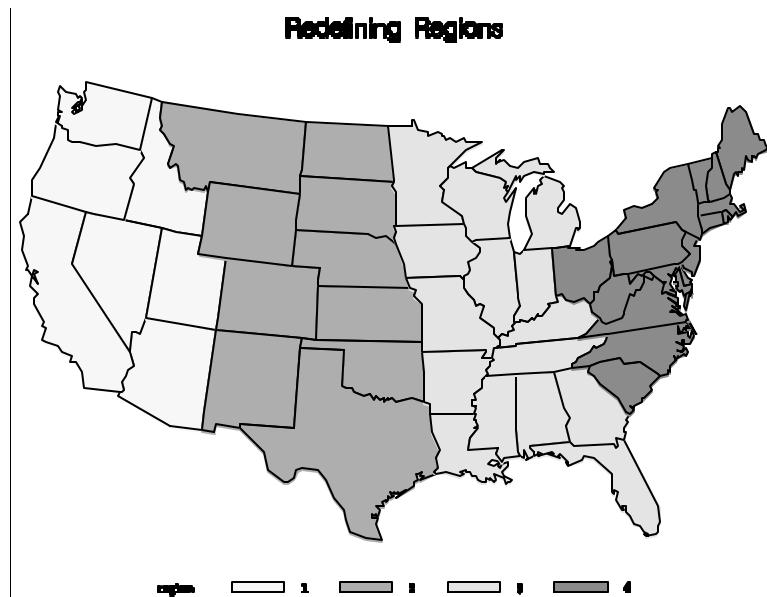
data votes;
  input stcode $ vote $ pop;
  state=stfips(stcode);
  cards;
CA Yes 123
ID Yes 240
UT Yes 350
TX Yes 100
NE Yes 250
MN Yes 370
IL No 380
WI No 340
GA No 160
VA No 255
VA No 130
;
proc format;
  value pop 100-199='100-199'
    200-299='200-299'
    300-399='300-399';
run;

pattern1 v=msolid c=liol;
pattern2 v=msolid c=bro;
pattern3 v=msolid c=vlip;

title 'Mapping a Numeric Response Variable';

proc gmap map=maps.us data=votes all;
  id state;
  choro pop / discrete coutline=black;
  label pop='Population';
  format pop pop.;
run;
quit;
```

Click on map to view program.



Defining Regions

```
* Defining regions (GREMOVE);
* Color regions with different colors;

* MVS users should remove the asterisk in the LIBNAME statement;
* libname maps 'zabcrun.sas810.maps' disp=shr;

goptions reset=all ftext=centb ftitle=swiss border;

data contig;
  set maps.us;
  where state not in(2,15); *remove Hawaii and Alaska;
  select(state);
    when(53,41,16,6,32,49,4) region=1;
    when(30,56,8,35,48,38,46,31,20,40) region=2;
    when(27,19,29,5,22,55,17,26,18,21,47,28,1,13,12) region=3;
    otherwise region=4;
  end;
run;

*define new unit areas;
proc sort data=contig;
  by region state;
run;

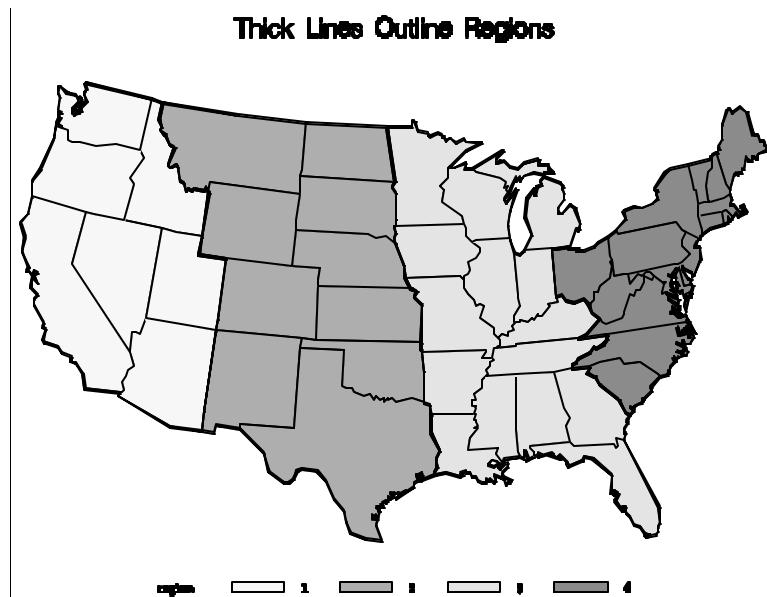
proc gremove data=contig out=contig2;
  id state;           * old unit areas;
  by region state;   * new unit areas;
run;

pattern1 v=msolid c=grayf8;
pattern2 v=msolid c=grayaf;
pattern3 v=msolid c=graye5;
pattern4 v=msolid c=gray8c;

title 'Redefining Regions';

proc qmap map=contig2 data=contig2;
  id region state;
  choro region / discrete coutline=black;
run;
quit;
```

Click on map to view program.



Thick Lines around Regions

```
* Defining regions (GREMOVE);
* Use thick lines for outline of regions;

* MVS users should remove the asterisk in the LIBNAME statement;
* libname maps 'zabcrun.sas810.maps' disp=shr;

goptions reset=all ftext=centb ftitle=swiss border;

*read coordinates of US (except Hawaii and Alaska) and create regions;
data contig;
  set maps.us;
  where state not in(2,15);
  select(state);
    when(53,41,16,6,32,49,4) region=1;
    when(30,56,8,35,48,38,46,31,20,40) region=2;
    when(27,19,29,5,22,55,17,26,18,21,47,28,1,13,12) region=3;
    otherwise region=4;
  end;
run;

*define new unit areas by REGION STATE;
proc sort data=contig;
  by region state;
run;

proc gremove data=contig out=contig2; *CONTIG2 is the new map data set;
  id state;           * old unit areas;
  by region state;   * new unit areas;
run;

*create annotate data set containing all coordinates that delineate
the regions and make the line thick using the annotate variable SIZE;
proc gremove data=contig2 out=contig3;
  id region state;      *removes state lines;
  by region;            *new unit areas is REGION;
run;

data outline;
  retain xsy sysys '2' when 'A' size 3 x1 y1; *necessary annotate variables;
  set contig3;
  by region;
  if first.region then do;      *first pt of region so start drawing lines;
    function='move';
    output;
    x1=x;y1=y;                 *keep track of first coordinates for region;
  end;
  else if last.region then do; *last pt of region so draw line to first pt;
    function='draw';
    output;
    x=x1; y=y1;                *first coordinates;
```

```
function='draw';
  output;
end;
else function='draw';output;
run;

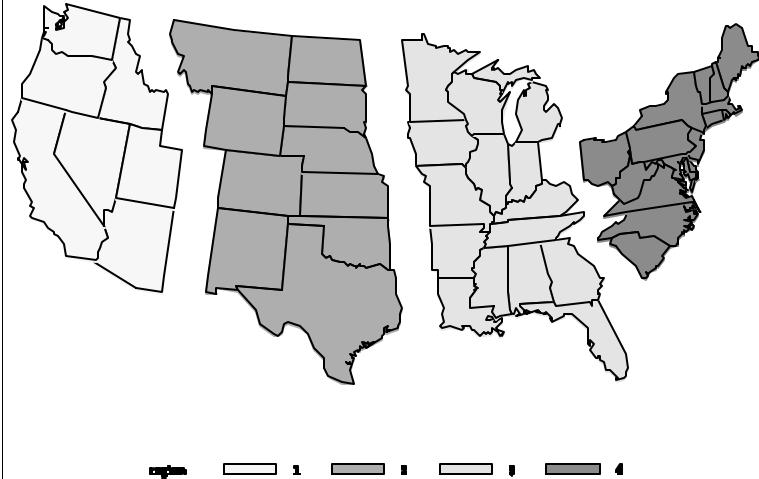
pattern1 v=msolid c=grayf8;
pattern2 v=msolid c=grayaf;
pattern3 v=msolid c=graye5;
pattern4 v=msolid c=gray8c;

title 'Thick Lines Outline Regions';

proc gmap map=contig2 data=contig2;
  id region state;
  choro region / discrete coutline=black anno=outline;
run;
quit;
```

Click on map to view program.

Redefining Regions with Regions Separated



Regions Separated

```
* Defining regions (GREMOVE);
* Separate regions so there's space between them;

* MVS users should remove the asterisk in the LIBNAME statement;
* libname maps 'zabcrun.sas810.maps' disp=shr;

goptions reset=all ftext=centb ftitle=swiss border;

data contig;
  set maps.us;
  where state not in(2,15); *remove Hawaii and Alaska;
  select(state);
    when(53,41,16,6,32,49,4) region=1;
    when(30,56,8,35,48,38,46,31,20,40) region=2;
    when(27,19,29,5,22,55,17,26,18,21,47,28,1,13,12) region=3;
    otherwise region=4;
  end;
run;

*define new unit areas;
proc sort data=contig; *needed for GREMOVE;
  by region state;
run;

proc gremove data=contig out=contig2; *new map data set is CONTIG2;
  id state;           * old unit areas;
  by region state;   * new unit areas;
run;

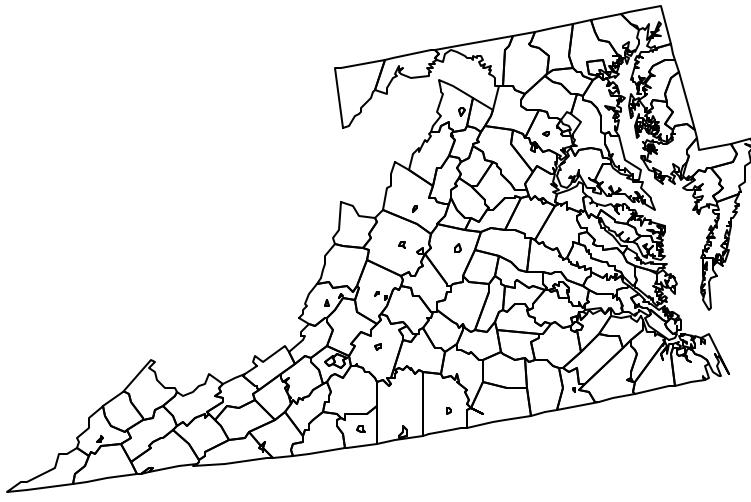
*shift values of X coordinate;
data explode;
  set contig2;
  select (region);
    when(1) x=x-.1;   *to left;
    when(2) x=x-.05;  *to left;
    when(4) x=x+.05;  *to right;
  otherwise;
    *region 3 stays where it was;
  end;
run;

pattern1 v=msolid c=grayf8;
pattern2 v=msolid c=grayaf;
pattern3 v=msolid c=graye5;
pattern4 v=msolid c=gray8c;

title 'Redefining Regions with Regions Separated';
proc gmap map=explode data=explode;
  id region state;
  choro region / discrete coutline=black;
run;
```

Click on map to view program.

Maryland and Virginia Counties



MD and VA Counties

```
* Counties;
* Maryland, Virginia (FIPS codes 24, 51);

* MVS users should remove the asterisk in the LIBNAME statement;
* libname maps 'zabcrun.sas810.maps' disp=shr;

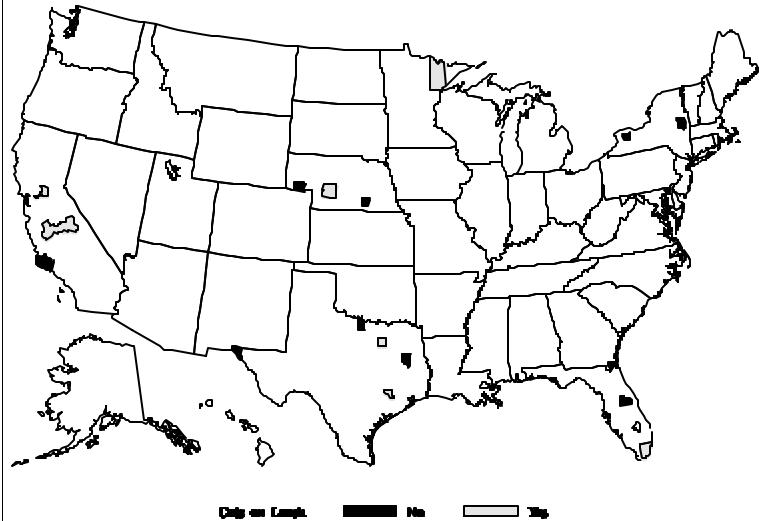
goptions reset=all ftext=centb ftitle=swiss border;
pattern v=me c=black r=150;

title 'Maryland and Virginia Counties';

proc gmap map=maps.uscounty(where=(state in(24,51)))
      data=maps.uscounty(where=(state in(24,51)));
  id state county;
  choro county / discrete nolegend;
run;
quit;
```

Click on map to view program.

Displaying all States with Some Counties



Drawing Some Counties Only

```
* Show all states but only some counties;
* MVS users should remove the asterisk in the LIBNAME statement;
* libname maps 'zabcrun.sas810.maps' disp=shr;

goptions reset=all ftext=centb ftitle=swiss border;

data catlaw;
  input stcode $2. @4 countynm && : $25. @20 catlaw $ 3.;
  countynm=upcase(countynm); *make it upper case since to match MAPS.CNTYNAME;
  state=stfips(stcode);
  label catlaw='Cats on Leash';
  cards;
CA Fresno      Yes
CA Sacramento  Yes
CA Santa Barbara No
CA San Luis Obispo No
TX Dallas       Yes
TX Austin        Yes
TX Cherokee     No
TX Clay          No
TX El Paso      No
NE Lincoln      Yes
NE Cheyenne     No
NE Clay          No
MN St Louis    Yes
FL Dade          Yes
FL Nassau        No
FL Orange        No
NY Genesee      No
NY Saratoga     No
;

* Need county code not name. Get from the MAPS.CNTYNAME data set (var COUNTY);
proc sort data=catlaw; *needed for MERGE;
  by state countynm;
run;

proc sort data=maps.cntyname out=cntyname; *needed for MERGE;
  by state countynm;
run;

data catlaw;           *response data set;
  merge catlaw(in=cat) cntyname;
  by state countynm;
  if cat;           *only keep obs for counties of interest;
  *REGION defines new unit areas. Later will draw the regions not
  the counties since we don't want all counties;
  region=county;
run;
```

```

* Get coordinates for these counties and for the states and define new
  unit area REGION. Set to -1 if county does not appear in CATLAW;
proc sort data=catlaw; *needed for MERGE;
  by state county;
run;

data catmap0;
  merge catlaw(in=cat) maps.uscounty;
  by state county;
  if not(cat) then region=-1; *all counties we don't want to delineate;
run;

* Remove old unit areas and define new ones based on STATE and REGION;
proc sort data=catmap0; *needed for GREMOVE;
  by state region;
run;

proc gremove data=catmap0 out=catmap; *CATMAP is now the map data set;
  id state county;
  by state region;
run;

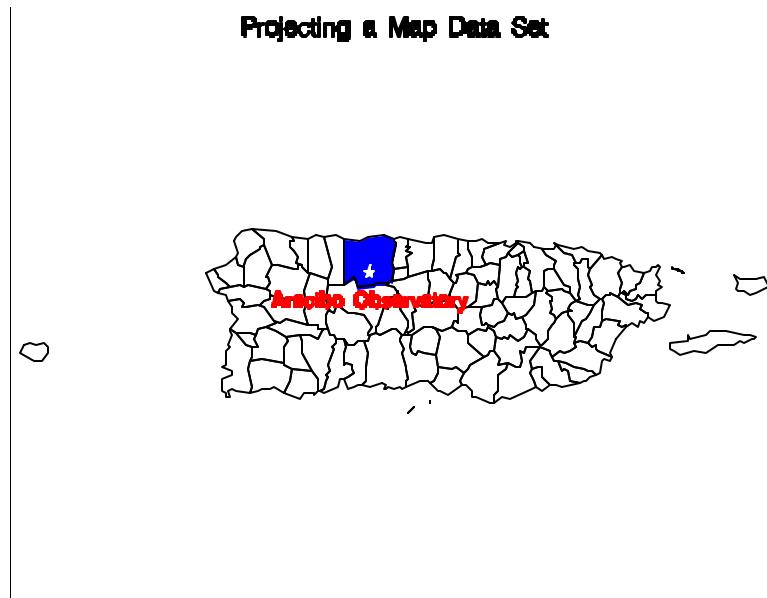
* draw map;
pattern1 v=ms c=blue;
pattern2 v=ms c=yellow;

title 'Displaying all States with Some Counties';

proc gmap map=catmap data=catlaw all;
  id state region;
  choro catlaw / coutline=black;
run;
quit;

```

Click on map to view program.



Projecting and Reducing a Map Data Set

```
* Projecting and Reducing a Map Data Set (GPROJECT, GREDUCE);
* Puerto Rico (FIPS code 72);

* MVS users should remove the asterisk in the LIBNAME statement;
* libname maps 'zabcrun.sas810.maps' disp=shr;

goptions reset=all border;

* Extract data from MAPS.COUNTIES and project it;
proc gproject data=maps.counties(where=(state=72)) out=pr;
  id state county;
run;

* Use GREDUCE to create the variable DENSITY used to reduce
  the data in the GMAP step;
proc greduce data=pr out=pr2;
  id county;
run;

* Create annotate data set for labels:
  1st obs = star, 2nd obs = character string;
data naic;
  retain xsyss ysys '2' when 'A' function 'label' size 1.6 ;
  length text $ 40 color $ 8 position $ 1;
  input style $ text $ && color $ position $ x y;
  cards;
special M white 5 -0.0015 0.00325
swissb Arecibo Observatory cxff0000 8 -0.0015 0.00325
;

title 'Projecting a Map Data Set';

pattern1 v=ms c=blue;

proc gmap data=pr2(where=(county=13))
  map=pr2(where=(density<=3)) all;
  id county;
  choro county / discrete nolegend coutline=black anno=naic;
run;
quit;
```

Click on map to view program.



US Cities

```
* Labelling Maps;
* Cities in US;

* MVS users should remove the asterisk in the LIBNAME statement;
* libname maps 'zabcrun.sas810.maps' disp=shr;

goptions reset=all ftext=centb ftitle=swiss border;

data baseball;
length city $ 80;
input stcode $ 1-2 city $ 5-13;
state=stfips(stcode); *need FIPS codes;
cards;
NY New York
MA Boston
GA Atlanta
IL Chicago
OH Cleveland
CA San Diego
TX Houston
TX Dallas
;

proc sort data=baseball;
by state city;
run;

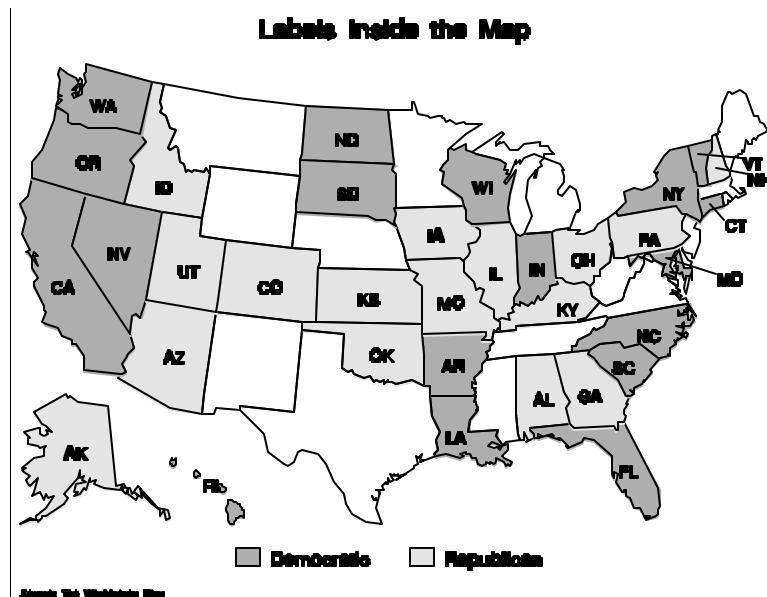
* Create annotate data set using MAPS.USCITY;
data cities;
retain xsyss ysys '2' when 'A'; *necessary annotate variables;
length text $ 20;
merge baseball(in=b) maps.uscity;
by state city;
if b; *only keep cities in our data set;
function='label'; *draw a label;
text='M'; *draw the character corresponding to M;
style='special'; *of the font SPECIAL (a star);
size=1.5; *in size 1.5;
position='5'; *right on top of coordinate;
color='cff0000'; *in bright red;
output;
function='label'; *draw a label;
text=city; *the city name;
style='swissb'; *using font SWISSB;
size=.9; *in size 1.2;
position='8'; *centered below coordinate;
color='black'; *in black;
x=x+.01; *shift a little to the right;
output;
run;
```

```
pattern v=mempty r=100;

title 'US Cities';

*want all states except Hawaii and Alaska;
proc gmap map=maps.us(where=(state not in(2,15))) data=maps.us;
  id state;
  choro state / discrete nolegend coutline=black anno=cities;
run;
quit;
```

Click on map to view program.



Labels Inside the Map

```
* Labelling Inside the Map;

* MVS users should remove the asterisk in the LIBNAME statement;
* libname maps 'zabcrun.sas810.maps' disp=shr;

goptions reset=all ftext=centb ftitle=swiss border;

* Read Senate data;
data senate;
  input stcode $ @@;
  if _n_ le 18 then party='D'; *first 18 Democratic;
  else party='R'; *rest Republican;
  state=stfips(stcode);
cards;
WA OR NV CA ND SD WI IN AR LA VT NY CT MD NC SC FL HI
ID UT AZ AK CO KS OK IA MO IL OH KY AL PA GA NH
;

* create annotate data set;
data labels;
  link read;           *go to label READ below to read next obs;
  retain xsyss ysyss '2' when 'A' size 1.3;
  length function $ 8;
  if ocean='N' then do;      *label should be placed inside state;
    function='label';        *draw label;
    text=fipstate(state);   *the label is the state postal code;
    position='5';           *right on top of coordinate;
    output;
  end;
  else if ocean='Y' then do; *outside coord appears before inside coord;
    function='move';         *label should be placed outside, move there first;
    output;
    function='label';        *there, draw label;
    text=fipstate(state);   *label should be placed in ocean, move there first;
    position='6';           *to the right of coordinate;
    output;
    link read;              *state code in ocean;
    function='draw';output;  *get inside coord (next obs);
  end;
  return;
*only read necessary states;
read: set maps.uscenter(where=(state in(
53 41 6 38 46 55 18 5 22 50 36 9 24 37 45 12 15 16 49
4 2 8 20 40 19 29 17 39 21 1 42 13 33 32)));
run;

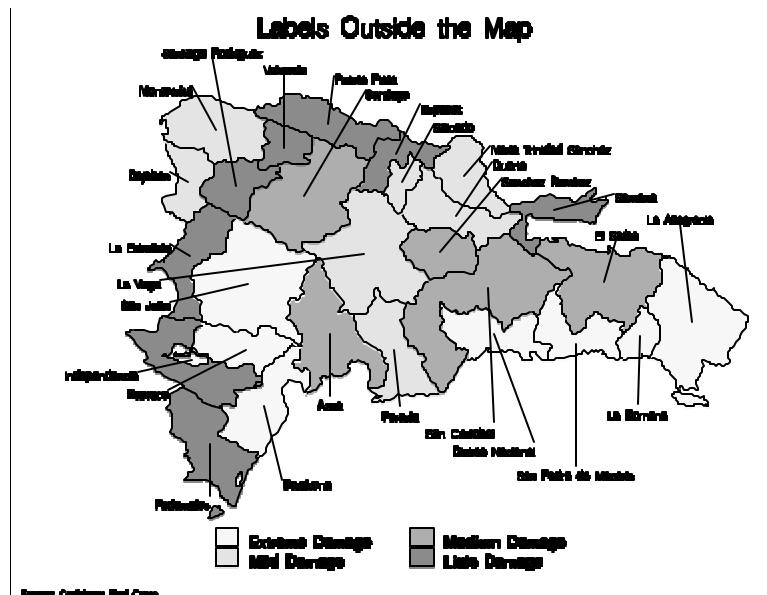
pattern1 v=ms c=blue;
pattern2 v=ms c=red;

title 'Labels Inside the Map';
footnote j=l h=.8 ' Source: The Washington Post';
```

```
legend1 value=(h=1.5 'Democratic' 'Republican')
      label=none shape=bar(3,3) pct;

proc gmap map=maps.us data=senate all;
  id state;
  choro party / discrete coutline=black anno=labels legend=legend1;
run;
quit;
```

Click on map to view program.



Labels Outside the Map

```
* Labelling Outside of Map;
* Draw map of the Dominican Republic with names of provinces;

* MVS users should remove the asterisk in the LIBNAME statement;
* libname maps 'zabcrun.sas810.maps' disp=shr;

goptions reset=all ftext=swiss border;

* create annotate data set;
data names;
  set maps.dominre2;
  retain xsys ysys '2' when 'A' size .9;
  *var POSITION already in data set;
  function='move ';           *move to;
  x=mid_x; y=mid_y;          *coordinate in visual center;
  output;
  function='draw ';           *draw line to;
  x=out_x; y=out_y;          *outside coordinate;
  output;
  function='label';           *there, write a label;
  text=idname;                *the label is the province name (var IDNAME);
  output;
  keep function when size xsys ysys x y text position id country;
run;

* Here we figure out area needed for labels in map since some go beyond
  the graphics area defined by outline of country;
proc means data=maps.dominre2 noprint;
  var out_x out_y;
  output out=max max=max_x max_y min=min_x min_y;
run;

* Extra obs that will be added to map data set;
data max;
  set max;
  country=320;
  segment=1;
  x=max_x; y=max_y; id=998; output;
  x=min_x-800; y=min_y; id=999; output;
run;

* Create new map data set by joining MAPS.DOMINREP and data set MAX
  and add response variable DAMAGE;
data dr;
  set maps.dominrep max;
  if      id in(2,3,5,10,12,22,23) then damage='1';
  else if id in(1,7,21,24,25) then damage='2';
  else if id in(4,6,13,14,15,17,19) then damage='3';
  else if id in(8,9,11,16,18,20,26,27) then damage='4';
run;
```

```
pattern1 v=ms c=red;
pattern2 v=ms c=blue;
pattern3 v=ms c=yellow;
pattern4 v=ms c=green;

title 'Labels Outside the Map';
footnote j=l h=.8 ' Source: Caribbean Red Cross';
legend label=none
      value=(h=1.2 f=swissb 'Extreme Damage' 'Medium Damage'
             'Mild Damage' 'Little Damage')
      across=2 shape=bar(3,3) pct;

proc gmap map=dr data=dr;
  id country id;
  choro damage / discrete anno=names coutline=black legend=legend1;
run;
quit;
```