CHAPTER 1

Introduction

1.1 Chapter Overview 2

1.2 Data Sets Needed to Produce a Map 2
1.2.1 Map Data Sets 2
1.2.2 Response Data Sets 4

1.3 Map Types in PROC GMAP 5
1.3.1 Choropleth Maps 5
1.3.2 Prism Maps 7
1.3.3 Block Maps 10
1.3.4 Surface Maps 15

1.4 Other Procedures That Use Map Data Sets 19
1.4.1 PROC GREMOVE 19
1.4.2 PROC GPROJECT 21
1.4.3 PROC GREDUCE 26

1.1 Chapter Overview

You can create maps with SAS by using PROC GMAP, one of the procedures available within SAS/GRAPH. Like other SAS procedures, PROC GMAP can be used on a number of levels. At a beginning level, you can produce a number of different types of maps using very little SAS code and no procedure options. At a more advanced level, you can create maps with labeled areas and hyperlinks to other information. This chapter introduces basic topics: the data needed to create maps, the various types of maps that can be produced, and some other SAS/GRAPH procedures that complement PROC GMAP.

1.2 Data Sets Needed to Produce a Map

You produce maps with PROC GMAP by using a combination of a map data set and a response data set. A map data set contains the information needed to draw map boundaries. Some typical map boundaries are defined by country, state, county, census tract, and zip code. A response data set contains the information that is to be displayed on the map, such as country-specific birth rates or state-specific populations.

The work that PROC GMAP performs is analogous to that performed by a match-merge in a DATA step. A match-merge combines observations from two or more data sets using one or more BY variables to match like observations. PROC GMAP matches observations from a response data set to those in a map data set using one or more ID variables. A DATA step match-merge results in a new data set, while PROC GMAP results in a geographical display of your response data.

1.2.1 Map Data Sets

In addition to giving you the graphics procedures that allow you to work with geographic-based data, SAS/GRAPH also provides map data sets that allow you to create at least one map of nearly every country in the world. The map data sets provided with SAS/GRAPH contain geographic areas (boundaries) represented in terms of longitude and latitude. At a minimum, the map data sets contain three variables: the two coordinates for the points of area boundaries, and a variable that contains a value for the geographic area associated with each set of coordinates. The variables that contain the coordinates used by PROC GMAP to draw maps must be numeric and must be named X (longitude) and Y (latitude). The variable that contains the value of the geographic areas is referred to as the identification variable. Its name varies among the map data sets; in some map data sets, a given set of coordinates may be associated with more than one geographic area (such as a state and a county). The map data sets are stored in the MAPS library, located in the directory MAPS under the SAS root directory. You can use the libref MAPS without having to use a LIBNAME statement. You can use any of the map data sets supplied by SAS in PROC GMAP by specifying a two-level data set name, such as MAPS.US or MAPS.USCOUNTY.

SAS/GRAPH provides several map data sets that you can use to draw maps of the United States. The US data set contains the following variables:

VARIABLE	TYPE	LABEL
SEGMENT	Num	State Segment Number
STATE	Num	State FIPS Code
X	Num	X Coordinate
Y	Num	Y Coordinate

In this data set, the variable STATE is the geographic area associated with the mapping coordinates. SEGMENT is another variable in this data set that is common to many of the map data sets. That variable is used by PROC GMAP to draw geographic areas that may comprise more than one polygon and still identify the multiple polygons as representing only one area. An example of such an occurrence in the US data set is the state of Hawaii, which is made up of several islands.

You can also use the USCOUNTY data set to draw a map of the United States. It is an example of a data set in which each set of coordinates is associated with more than one geographic area. The data set contains the following variables:

VARIABLE	TYPE	LABEL
COUNTY	Num	County FIPS Code
SEGMENT	Num	County Segment Number
STATE	Num	State FIPS Code
X	Num	X Coordinate
Y	Num	Y Coordinate

The addition of the variable COUNTY enables you to draw maps of the United States that show both state and county boundaries. Additional variables appear in some of the other map data sets. They are discussed later in this chapter, as are attributes of the X-Y coordinate pairs that vary among the map data sets.

1.2.2 Response Data Sets

A response data set contains the data that you want to display in the form of a map. Just as each map data set contains a variable that associates each set of coordinates with a geographic area, each observation to be displayed in a map must also contain a variable that can be used to match it to an area in the map data set. This common variable must have the same name and be of the same type as the variable in the map data set. If the US map data set is being used to draw a map, the numeric variable STATE identifies areas. To display data associated with any of the states, a response data set must also contain a numeric variable named STATE that allows PROC GMAP to match the data to be displayed with the correct map areas.

The response data set (US2000ST) used in most of the examples contains four variables:

VARIABLE	TYPE	LABEL
POP1990	Num	Year 1990 Census Population
POP2000	Num	Year 2000 Census Population
REGION	Char	Census Region
STATE	Num	State FIPS Code

The data set can be created using SAS code and data found in Appendix A1. That SAS code creates a temporary SAS data set, placing it in the WORK library. All the examples assume that this data set is in the WORK library. If you want to create a permanent SAS data set by modifying the code in Appendix A1.1 as follows, then you must also modify the examples to explicitly state the location of the data set US2000ST, a libref other than WORK.

```
libname pop 'd:\census';
data pop.us2000st;
<rest of SAS code>
```

The values of the variable STATE in a response data set must correspond to the values of that variable in the US map data set. The FIPNAME function, supplied by SAS, can be used to display the state name associated with any given FIPS code. (FIPS stands for Federal Information Processing Standards.) SAS code in Appendix A2 shows an easy way to list numeric FIPS codes and names.

1.3 Map Types in PROC GMAP

You can create four different map types using PROC GMAP: choropleth, prism, block, and surface. Your choice of map type depends on the information you want to convey. In some situations, more than one type of map may be appropriate. In others, a given map type may actually hinder understanding of your data. Each of these map types is shown in the following series of examples. The data displayed in each map are the year 2000 census state-specific populations of the United States (data was obtained from the U.S. Census Bureau Web site). As stated in section 1.2, the response data set, US2000ST, to be used in the examples is assumed to be in the WORK library and can be created using the SAS code and data shown in Appendix A1.

1.3.1 Choropleth Maps

A choropleth map uses shading, patterns, or colors to distinguish map areas (in this case, states). In Figure 1.1, four different population levels are shown using shading, with darker shading indicating a higher population. The four levels represent quartiles, or the 51 areas (50 states plus the District of Columbia) divided into four groups of approximately 13 states each.

Example 1.1 Create a choropleth map

```
proc format;  ❶
value pop
low     -< 1300000 = '<1.3'     1300000 -< 4000000 = '1.3-3.9'
4000000 -< 6000000 = '4.0-5.9'  6000000 -  high    = '6.0+'
;
run;
pattern1 v=ms c=grayfa;  ❷
pattern2 v=ms c=grayda;
pattern3 v=ms c=grayaa;
pattern4 v=ms c=gray5a;

title 'YEAR 2000 CENSUS POPULATION';

proc gmap  ❸
map=maps.us
data=us2000st;
id state;  ❹
choro pop2000  ❺ / discrete coutline=black;  ❻
label pop2000='MILLIONS';  ❼
format pop2000 pop.;  ❽
run;
quit;
```

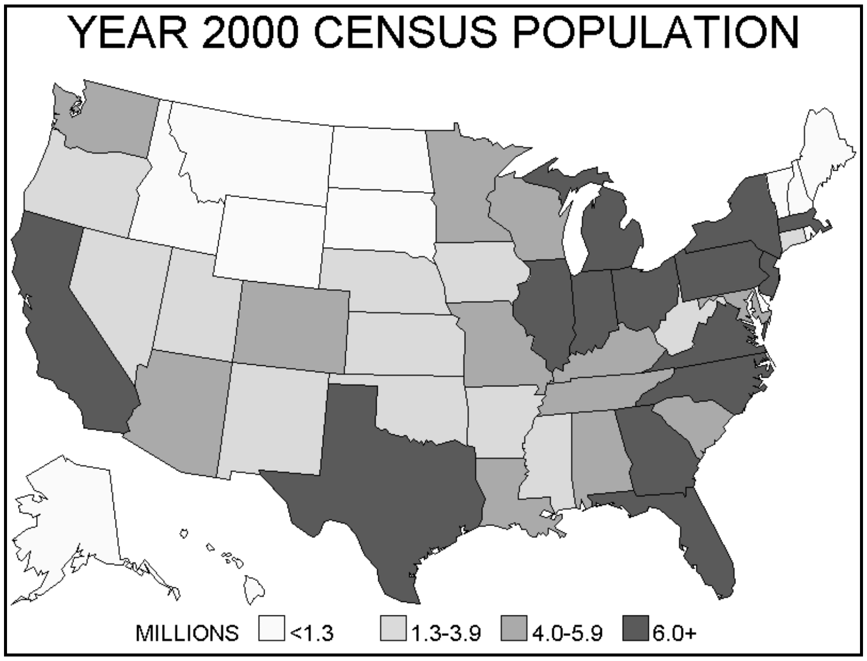

Figure 1.1 *Choropleth map*

This example creates a format, POP,❶ that will be used to group states based on their year 2000 population. Using a format to group geographic areas in a map is no different from grouping observations in the output from other non-graphic procedures. Just as you might use a format to group observations in a table produced using PROC TABULATE, you can use a format to group geographic areas when using PROC GMAP.

You can use a number of methods to represent the four levels of population in the map. One method is to use different hatching patterns—lines drawn within areas at various angles and spacings. Another method is to use color. The method used here is to assign a different gray-scale value to each of four population levels.❷ Patterns have two main attributes: a value, defined here as V=MS, or VALUE=MSOLID; and a color, defined here as C=GRAYxx, or COLOR=GRAYxx. The gray scale in SAS has 256 levels. The xx values following the word GRAY are hexadecimal values that range from 00 (gray00) or black (no color) to FF (grayff) or white (all colors combined; remember what happens when you pass white light through a prism). The four colors shown in the PATTERN statements start at a value close to white (grayfa) and progress to a value closer to black (gray5a).

The US map data set supplied by SAS and the response data set containing the population data are used to create a map.❸ The variable that is common to the two data sets is STATE. If you were match-merging two data sets, you would tell SAS to match observations according to the variable(s) in a BY statement. In PROC GMAP, an ID statement ❹ declares the variable(s) that matches response data set observations to map data set areas.

A CHORO statement ❺ instructs PROC GMAP to create a choropleth map displaying values of the variable POP2000 in the response data set. Two options in the CHORO statement ❻ override the default behavior of PROC GMAP. First, the DISCRETE option treats the response variable (POP2000) as having distinct levels rather than as a continuous variable. The levels are defined in the user-written format POP. Without the DISCRETE option (or the LEVELS option), a formula (shown in PROC GMAP documentation) determines the number of levels used to display values of the response variable. Second, the COUTLINE option results in a map with states outlined in black. Without the COUTLINE option, areas are outlined in the colors specified in the PATTERN statements used to fill map areas. This default behavior may result in map areas that cannot be distinguished from each other if they lie next to each other and have the same color.

PROC GMAP uses the label of the response variable to label the map legend. If no label is present, the name of the variable is used. The LABEL statement ❼ changes the legend label from that in the response data set to "millions." The FORMAT statement ❽ groups the populations in the response data set according to the ranges specified by the format POP.

1.3.2 Prism Maps

In addition to the shading, patterns, and colors used by choropleth maps, a prism map uses the height of raised map areas to convey information. The height of the map areas is proportional to the ordinal level (rank) of the data values in the response data set. Only one change is needed in the SAS code of Example 1.1 to create a prism map instead of a choropleth map. You use the same FORMAT, PATTERN, and TITLE statements. The change is made within PROC GMAP, where the CHORO statement in Example 1.1 has been replaced by a PRISM statement.❶

Example 1.2 Create a prism map

```
proc format;
value pop
low     -< 1300000 = '<1.3'     1300000 -< 4000000 = '1.3-3.9'
4000000 -< 6000000 = '4.0-5.9'  6000000 -  high    = '6.0+'
;
run;

pattern1 v=ms c=grayfa;
pattern2 v=ms c=grayda;
pattern3 v=ms c=grayaa;
pattern4 v=ms c=gray5a;

title 'YEAR 2000 CENSUS POPULATION';

proc gmap
map=maps.us
data=us2000st
;
```

```
id state;
prism pop2000 ❶  / discrete coutline=black;
label pop2000 = 'MILLIONS';
format pop2000 pop.;
run;
quit;
```

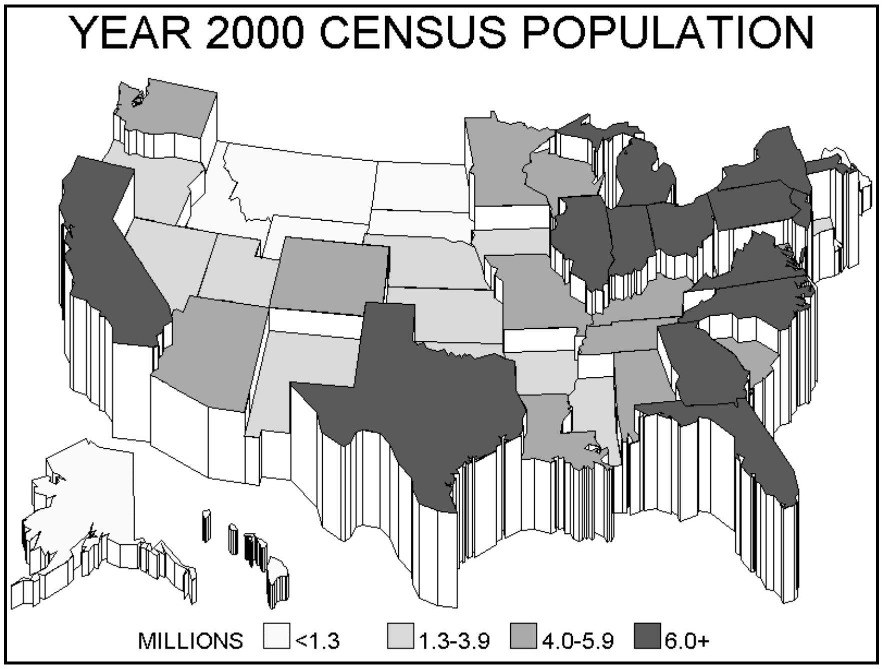

Figure 1.2 *Prism map*

The default map produced by the PRISM statement in this example may obscure some map areas behind other raised areas. You can use several options to alter the viewing angle of a prism map. The default viewing angle is a position above and to the south of the center of the map. You can change three features of the viewing angle using XVIEW (east-west location, default at the map center, 0.5), YVIEW (north-south location, default south of the map, –2.0), and ZVIEW (height above the map, 3.0). The PROC GMAP documentation contains a detailed explanation of the X-Y-Z coordinate system and the X-Y-Z views. Example 1.3 shows the change in the appearance of the prism map if the YVIEW (moved north) and ZVIEW (moved higher) are altered. ❶ Each prism map also has an imagined light source, whose position controls the appearance of the raised edges of the map areas. You can alter the position of the light source using the XLIGHT and YLIGHT options. The map shown in Figure 1.3 contains shading that has been added to the edges of all the map areas by changing the YLIGHT value.❷

Example 1.3 Create a prism map with several options

```
proc format;
value pop
low     -< 1300000 = '<1.3'
1300000 -< 4000000 = '1.3-3.9'
4000000 -< 6000000 = '4.0-5.9'
6000000 -  high    = '6.0+'
;
run;

pattern1 v=ms c=grayfa;
pattern2 v=ms c=grayda;
pattern3 v=ms c=grayaa;
pattern4 v=ms c=gray5a;

title 'YEAR 2000 CENSUS POPULATION';

proc gmap
map=maps.us
data=us2000st
;
id state;
prism pop2000 / discrete coutline=black yview=-0.5 zview=4.0 ❶  ylight=2 ❷;
label pop2000 = 'MILLIONS';
format pop2000 pop.;
run;

quit;
```

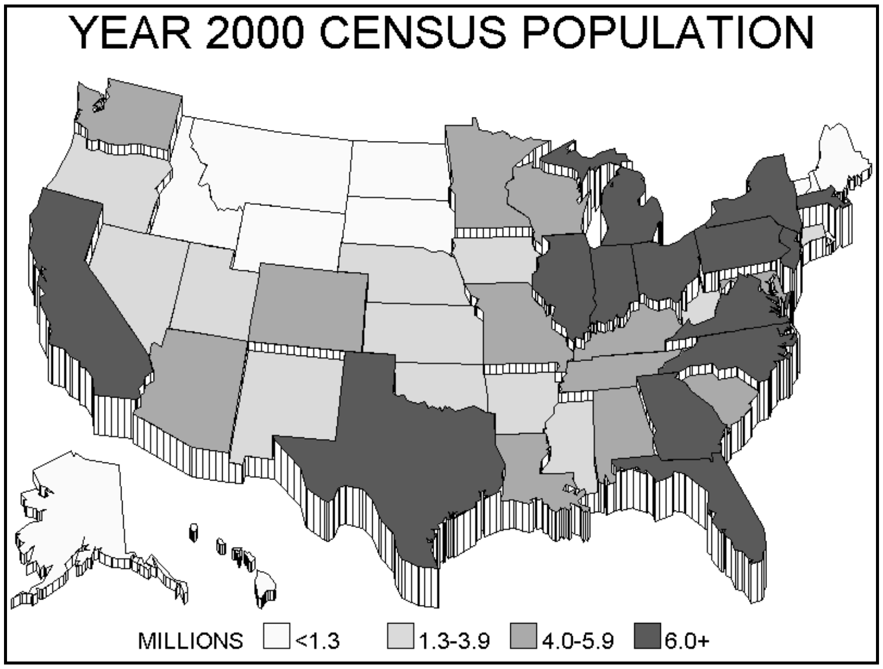

Figure 1.3 *Prism map with modified view and light source*

1.3.3 Block Maps

The block map combines features of choropleth and prism maps. You can distinguish among the map areas because the area boundaries are displayed. However, the map areas are not shaded based on data values. Instead, a block is placed at the center of each map area and the height of that block is the ordinal level (rank) of the response variable. The blocks are also shaded, as with prism maps, where both shading and the height of raised areas convey information about the response variable.

Example 1.4 Create a block map

```
proc format;
value pop
low     -< 1300000 = '<1.3'
1300000 -< 4000000 = '1.3-3.9'
4000000 -< 6000000 = '4.0-5.9'
6000000 -  high    = '6.0+'
;
run;

pattern1 v=s c=grayfa; ❶
pattern2 v=s c=grayda;
pattern3 v=s c=grayaa;
pattern4 v=s c=gray5a;

title 'YEAR 2000 CENSUS POPULATION';

proc gmap
map=maps.us
data=us2000st
;
id state;
block pop2000 ❷  / discrete coutline=black cblkout=black ❸ ;
label pop2000 = 'MILLIONS';
format pop2000 pop.;
run;
quit;
```

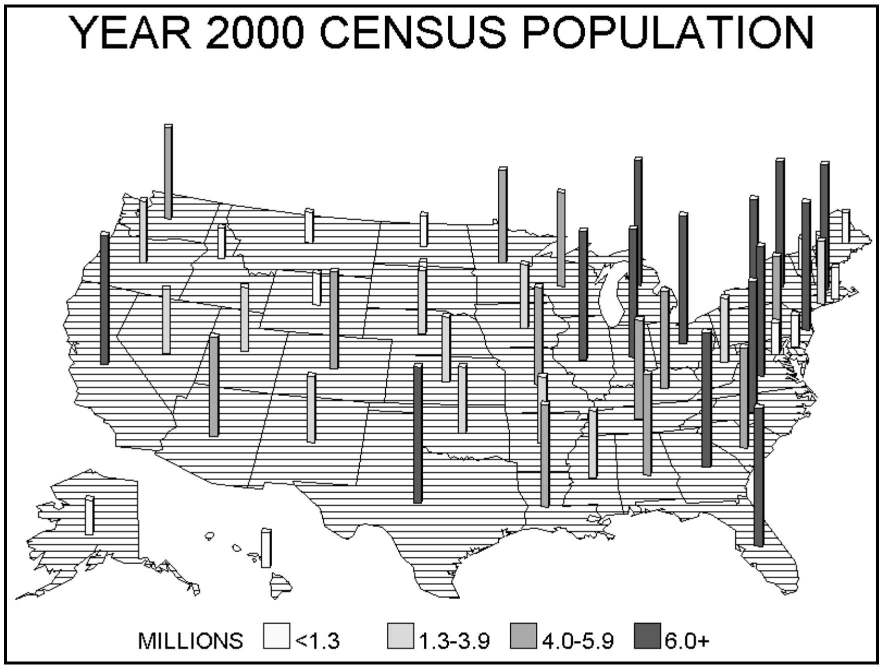

Figure 1.4 *Block map*

In this example, the BLOCK statement ❷ creates a map with raised blocks positioned at the center of each map area. The blocks are shaded with grays. The PATTERN statements ❶ are specified as V=S. A value of MS as used in the previous examples controls the fill for map areas, not for the raised blocks. The DISCRETE option plus the FORMAT statement results in blocks drawn at four different heights, proportional to the ordinal level (rank) of the formatted values of the response variable, POPULATION. Blocks are outlined in black ❸ as a result of using the CBLKOUT (color of block outline) option. The fill pattern for the map areas is equally spaced horizontal lines. If no PATTERN statements had been used to specify the fills of the blocks, the map areas would have used the default solid fill.

As with the prism map, you can use options to alter the appearance of the block map. Several of these options are shown in Example 1.5.

Example 1.5 Create a block map with several options

```
proc format;
value pop
low     -< 1300000 = '<1.3'
1300000 -< 4000000 = '1.3-3.9'
4000000 -< 6000000 = '4.0-5.9'
6000000 -  high    = '6.0+'
;
run;

pattern1 v=s c=grayfa;
pattern2 v=s c=grayda;
pattern3 v=s c=grayaa;
pattern4 v=s c=gray5a;
pattern5 v=ms c=grayea; ❶

title 'YEAR 2000 CENSUS POPULATION';

proc gmap
map=maps.us
data=us2000st
;
id state;
block pop2000 / discrete coutline=black cblkout=black blocksize=6 ❷
shape=prism ❸
                xview=0.75 yview=-1.0 zview=3.5 ❹ ;
label pop2000 = 'MILLIONS';
format pop2000 pop.;
run;
quit;
```

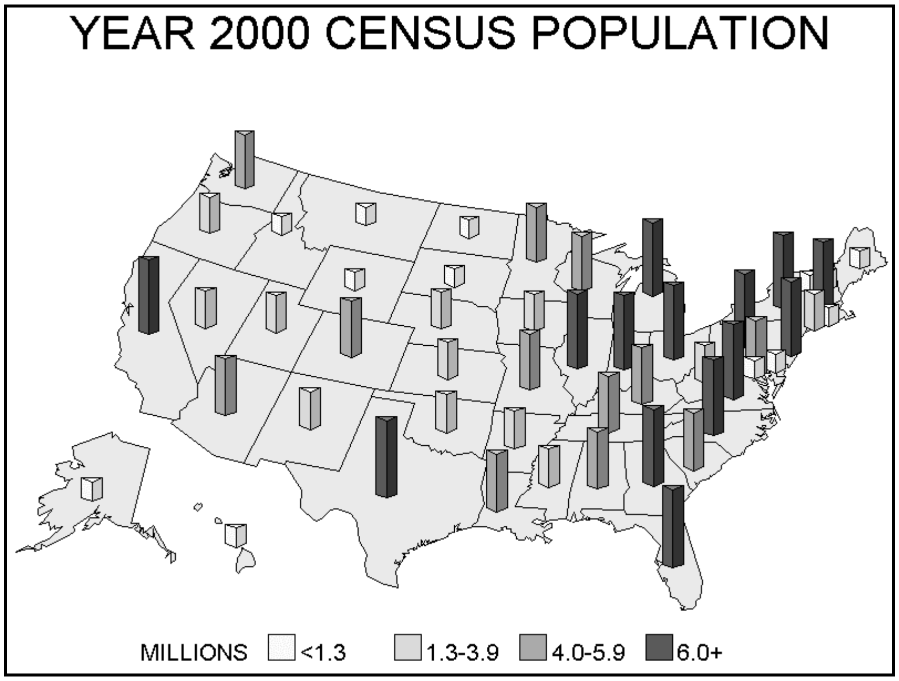

Figure 1.5 *Block map with modified view, block width, and area fill pattern*

This example uses a fifth PATTERN statement to control the fill used for map areas. ❶ Two PATTERN attributes are defined. The value (V) of the PATTERN is MS, not S as was used for the block fill patterns. The prefix M, for map, indicates that this PATTERN is to be used for the map areas. If you wanted an empty pattern, you would use ME to distinguish between a map area pattern and a block fill pattern. The width of the blocks is changed to 6 (from the default value of 2) using the BLOCKSIZE option, ❷ and their SHAPE ❸ is changed from the default block to a prism using the SHAPE= option. The default view is also modified ❹ (as was done with the prism map in Example 1.3). The default view values are the same as with the prism map. Changing the XVIEW to 0.75 from the default 0.50 moves the viewpoint to the east. A YVIEW value of –1.0 moves the viewpoint north from the default –2.0, which lies to the south of the map. Finally, changing the ZVIEW to 3.5 from the default 3.0 moves the viewing position higher over the map.

Rather than displaying discrete values of the response variable, a block map can display blocks used to show relative values of the response variable—that is, taller blocks have higher populations. The map is not intended to allow a viewer to assign a value of the response variable to a given map area.

Example 1.6 Create a continuously scaled response variable

```
pattern1 v=s  c=gray8a r=51; ❶
pattern2 v=ms c=grayea;
title "YEAR 2000 CENSUS POPULATION";

proc gmap
map=maps.us
data=us2000st
;
id state;
block pop2000 / coutline=black nolegend ❷  levels=51 ❸
                blocksize=6 shape=cylinder ❹ ;
run;
quit;
```

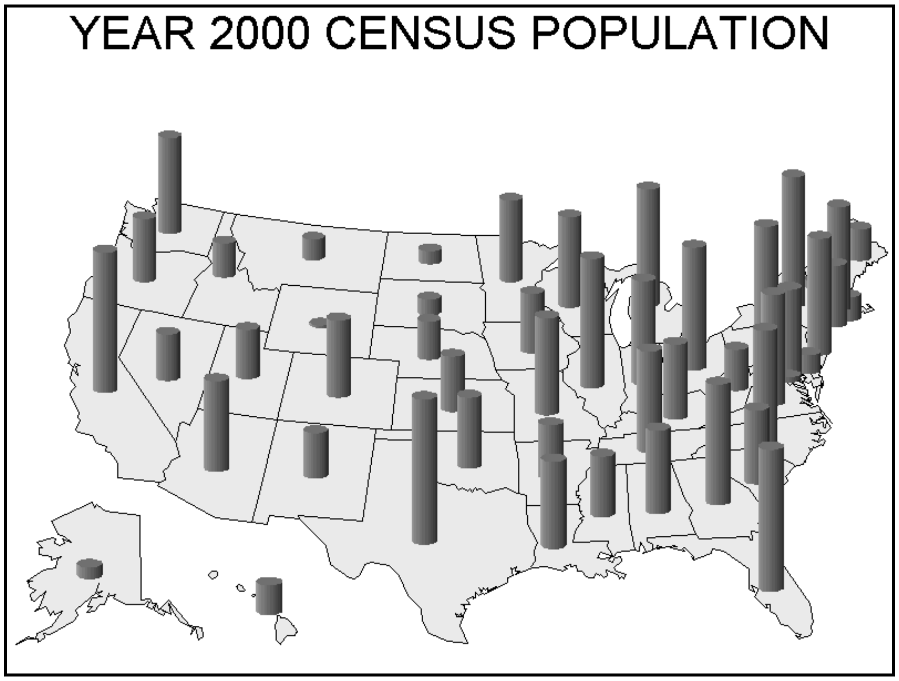

Figure 1.6 *Block map with continuously scaled blocks*

This example uses one pattern for the 51 blocks in the map ❶ by including the R (REPEAT) option in the PATTERN statement. The legend is suppressed ❷ since there is no use for a legend in this type of display. The LEVELS option results in blocks whose heights are scaled continuously from the lowest to the highest value (51 values—one per state plus the District of Columbia) of the response variable. ❸ The SHAPE option specifies the shape of the cylinder chosen for the raised areas. ❹

1.3.4 Surface Maps

The surface map is a departure from the other types of maps in that it does not display the boundaries of map areas. Rather, it uses a spike at the center of each map area, and the heights of the spikes distinguish among values of the data being displayed. The heights of the spikes show relative, not exact, values of the response variable. The choropleth, prism, and block map examples use the DISCRETE option and a format to create four groups of the response variable, POPULATION, and a map with four different gray-scale shadings of either map areas or blocks. A surface map scales the height of map spikes continuously from the lowest to the highest value of the response variable (similar to the block map in Example 1.6). No format is needed to group observations, and no PATTERN statements are needed since the map areas are not shaded.

Example 1.7 Create a surface map

```
title 'YEAR 2000 CENSUS POPULATION';

proc gmap
map=maps.us
data=us2000st
;
id state;
surface pop2000; ❶
run;
quit;
```

The SURFACE statement ❶ creates the map in Figure 1.7. No interior map areas are displayed.

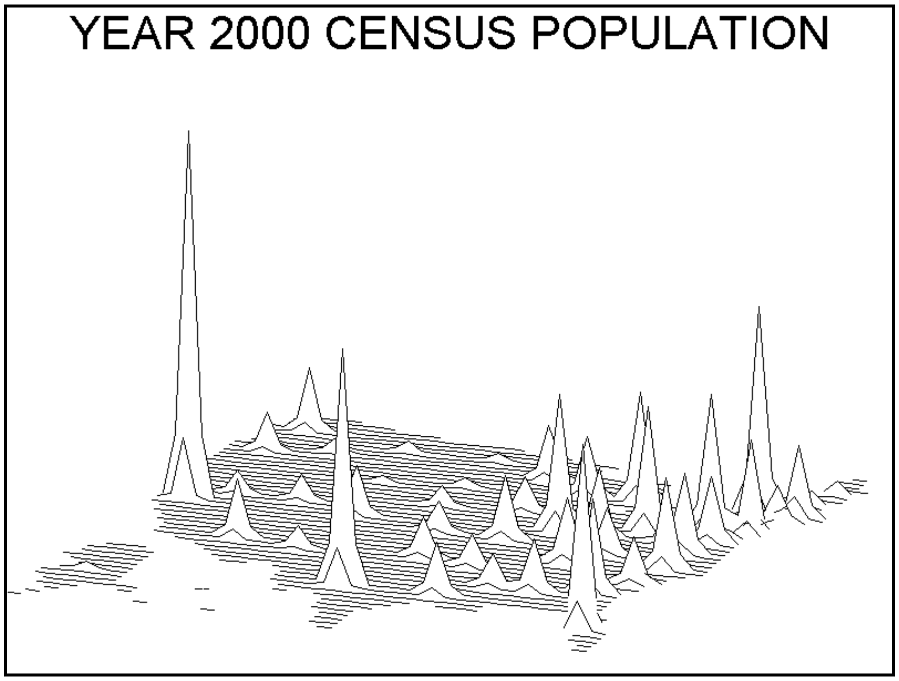

Figure 1.7 *Surface map*

The spikes rise from the center of the map areas to indicate the relative populations of the states. The block map in Figure 1.6 presents a similar display of the population data, created using a LEVELS option. No LEVELS option is available (or needed) when creating a surface map. The intent of a surface map is to show relative values of the response data, and no discrete display is possible.

Several options can alter the appearance of a surface map. The viewing angle is controlled by two options, ROTATE and TILT. The ROTATE option controls the position of the map with respect to the z axis, while the TILT option controls the position of the map with respect to the x axis. The default value of both options is 70 degrees. Higher values of ROTATE turn the map in a counterclockwise direction. TILT can vary from 0 to 90 degrees, or from directly overhead to an edge view, respectively.

Example 1.8 Create a surface map with altered viewing angle

```
title 'YEAR 2000 CENSUS POPULATION';

proc gmap
map=maps.us
data=us2000st
;
id state;
surface pop2000 / rotate=110 tilt=60 ❶ ;
;
run;

quit;
```

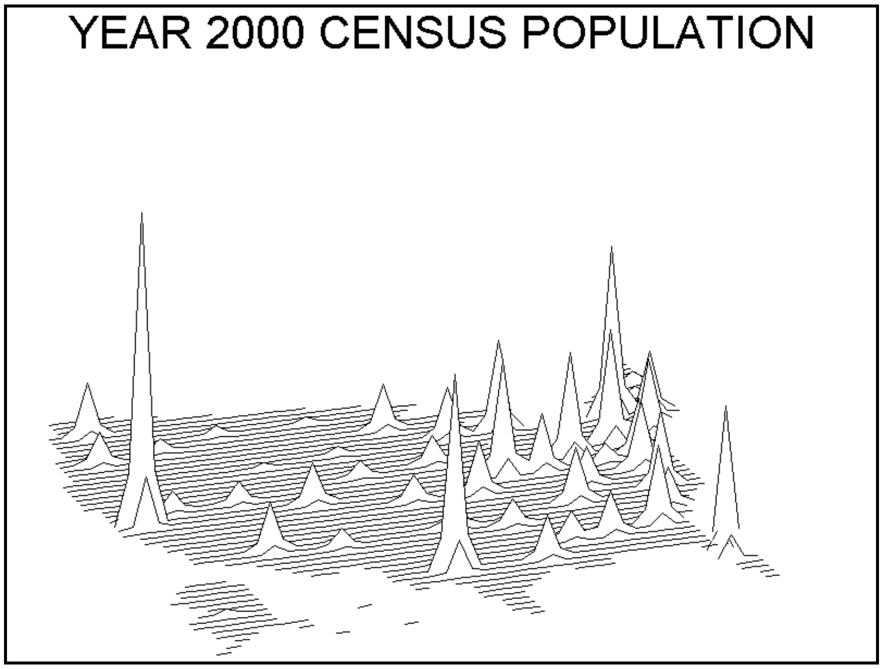

Figure 1.8 *Surface map with altered viewing angle*

In this example, rotating the map 110 degrees (40 degrees beyond the default value) has moved the view in the counterclockwise direction. ❶ The viewing point above the map has risen slightly, from the default of 70 degrees to 60 degrees (remember that 0 degrees is directly above the map).

You can use two other options to change the appearance of the map surface. The NLINES option controls the number of lines used to draw the map surface. The default value is 50, with an allowable range of 50 to 100. The CONSTANT option controls the appearance of the spikes. The default value is 10. Numbers greater than 10 result in spikes that are taller and have a wider base, and numbers less than 10 have the opposite effect.

Example 1.9 Create a surface map with altered viewing angle and map surface

```
title 'YEAR 2000 CENSUS POPULATION';

proc gmap
map=maps.us
data=us2000st
;
id state;
surface pop2000 / rotate=110 tilt=50 ❶  nlines=100 constant=20 ❷  ;
run;
quit;
```

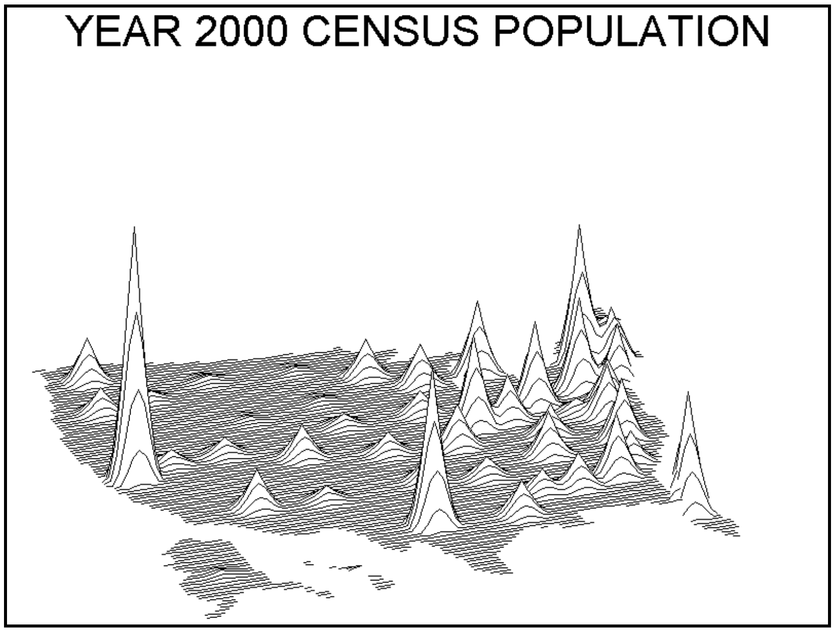

Figure 1.9 *Surface map with altered viewing angle and map surface*

This figure uses the same rotation as in Figure 1.8, but the tilt has been moved higher over the map. ❶ The surface of the map looks more dense. This was accomplished by using the maximum value of the NLINES option. ❷ The appearance of the spikes was also altered by doubling the value of the CONSTANT option from 10 to 20.

1.4 Other Procedures That Use Map Data Sets

Several other SAS/GRAPH procedures use map data sets. Their function is not to draw maps but to work with and modify the existing map data sets, creating new map data sets in the process. The procedures are discussed briefly in this chapter and more extensively in Chapter 3.

1.4.1 PROC GREMOVE

You can use PROC GREMOVE to remove the internal boundaries of a map area, combining already defined map areas (in this example, states) into a larger map area or areas. The US map data set used in all the examples thus far contains internal state boundaries within the mainland portion of the United States. If the state boundaries were removed, you could use PROC GMAP to draw an outline map of the United States.

In addition to a variable (STATE) that associates each X-Y coordinate in the data set with current map areas, the data set must also contain another variable that associates the coordinates with the new map area or areas. Since the US map data set does not contain a variable with a single value common to all observations in the data set, you can use a DATA step to add that variable to the data set.

Example 1.10 Create an outline map of the US mainland

```
data usa_mainland; ❶
retain country 'USA'; ❷
set maps.us;
where state not in (2,15); ❸
run;

proc gremove ❹
data=usa_mainland
out=usa_outline
;
id state; ❺
by country;
run;

pattern v=me c=black;

title h=6 'US MAP DATA SET - NO INTERNAL BOUNDARIES';

proc gmap
map=usa_outline ❻
data=usa_outline
;
```

```
   id country;
   choro country / nolegend;
   run;
   quit;
```

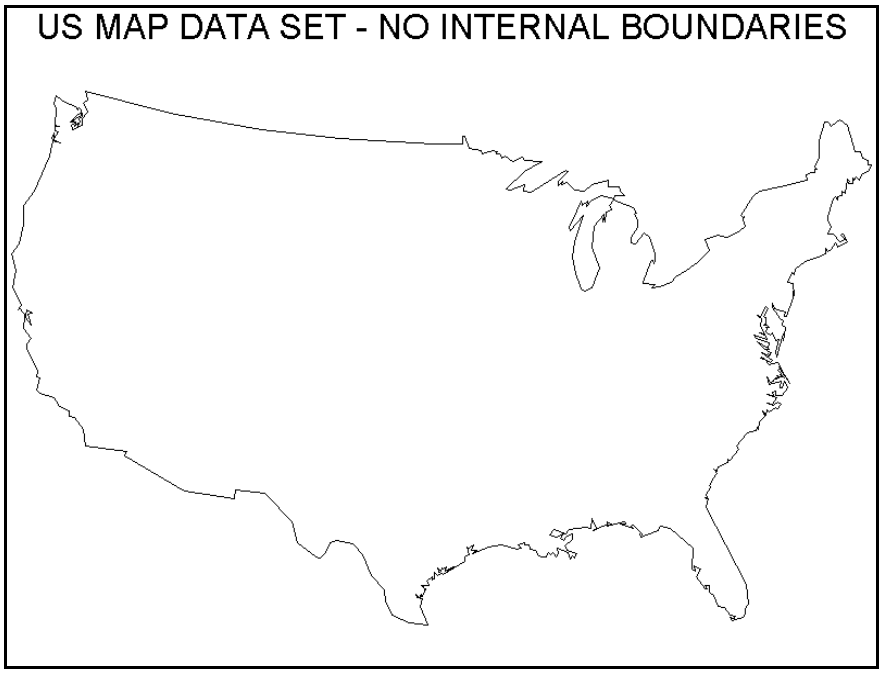

Figure 1.10 *Outline map of the US mainland (no Alaska or Hawaii)*

This example uses the US map data set to create a new map data set. ❶ A new variable, COUNTRY, is added to each observation in the new data set. ❷ The new variable will identify the new map areas for PROC GREMOVE. The map excludes observations for two states, Alaska (FIPS code 2) and Hawaii (FIPS code 15).❸ Two-character postal codes are probably more familiar than FIPS codes. You can use the FIPSTATE function to convert FIPS codes to two-character postal codes. You can use that function in the WHERE statement to exclude Alaska and Hawaii:

```
   where fipstate(state) not in ('AK','HI');
```

The FIPSTATE function is used in subsequent examples when FIPS code specification is required.

PROC GREMOVE removes the state boundaries from the new data set. ❹ The variable that defines old map areas (STATE) is used as the ID variable, ❺ while the variable that defines the new map area (COUNTRY) is the BY variable. The new data set is used as both the map data set and the response data set. ❻ The map in Figure 1.10 depicts the outline of the continental United States.

1.4.2 PROC GPROJECT

Understanding the function of the GPROJECT procedure requires a little more knowledge of map data sets. The US map data set used to produce all the maps shown thus far contains projected X-Y coordinates—projected longitude and latitude, respectively—with Alaska and Hawaii repositioned at the lower right of the other states. The process of converting longitude and latitude from a spherical coordinate system to a flat, two-dimensional plane is called projection. PROC GPROJECT is used to convert the X-Y coordinates from latitude and longitude to arbitrary Cartesian coordinates. Referring to the X-Y coordinates in the US map data set as *projected* means that they have undergone this conversion process.

The map data set STATES also contains X-Y coordinates for state boundaries and contains the following variables:

VARIABLE	TYPE	LABEL
DENSITY	Num	Density for Lower Resolution Maps
SEGMENT	Num	State Segment Number
STATE	Num	State FIPS Code
X	Num	Unprojected Longitude in Radians
Y	Num	Unprojected Latitude in Radians

Although you can use both the US and STATES map data sets to create a map of the United States, there are two main differences between these data sets. First, the STATES data set contains an additional variable, DENSITY (more about that in the section on PROC GREDUCE). Next, the X-Y coordinates are unprojected. Figure 1.11 shows a map that results if the STATES data set is used directly with PROC GMAP. The following example substitutes STATES in the SAS code that produced Figure 1.1.

Example 1.11 Create a choropleth map using unprojected STATES map data set

```
proc format;
value pop
low     -< 1300000 = '<1.3'      1300000 -< 4000000 = '1.3-3.9'
4000000 -< 6000000 = '4.0-5.9'   6000000 -  high    = '6.0+'
;
run;

pattern1 v=ms c=grayfa; pattern2 v=ms c=grayda;
pattern3 v=ms c=grayaa; pattern4 v=ms c=gray5a;

title 'YEAR 2000 CENSUS POPULATION';

proc gmap
map=maps.states
data=us2000st
;
```

```
id state;
choro pop2000 / discrete coutline=black;
label pop2000 = 'MILLIONS';
format pop2000 pop.;
run;
quit;
```

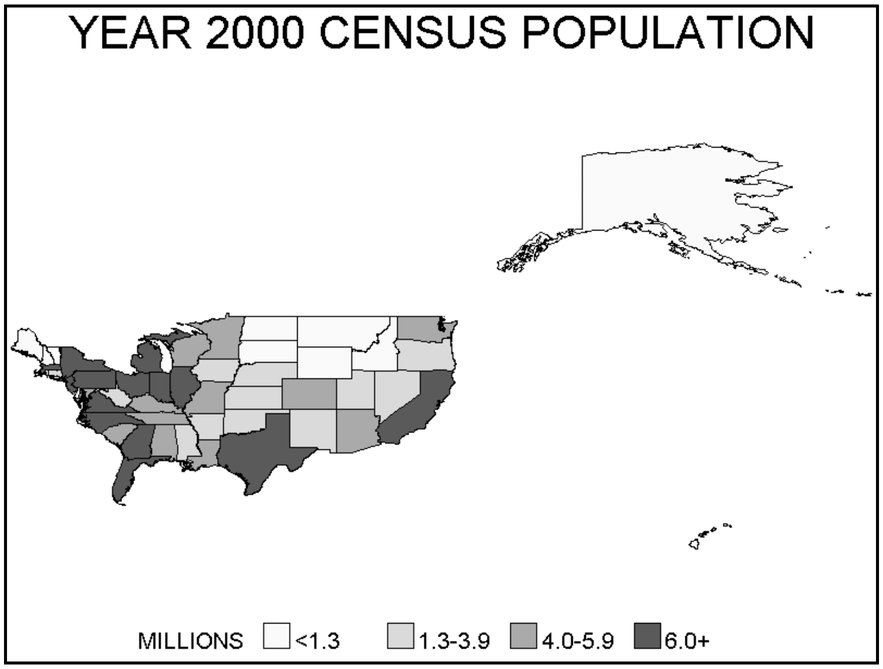

Figure 1.11 *Map created using unprojected STATES map data set*

There are several obvious differences between the map drawn with the projected US map data set and the map drawn with the unprojected STATES map data set. The unprojected map is drawn backwards and the shapes of all the map areas are distorted. Alaska and Hawaii are also now in approximately the correct position relative to the rest of the United States. The map areas face in the wrong direction since the X coordinate (longitude) in the map data set increases in size from east to west. PROC GMAP draws the map areas with an assumed origin in the lower-left corner and X coordinates increasing from left to right (west to east). The unprojected coordinates are measured on a sphere and have not been adjusted to fit on a flat surface. One difference that may not be obvious is the absence of Puerto Rico though the STATES map data set contains X-Y coordinates for Puerto Rico. Since the response data set did not provide a population for Puerto Rico, that map area was not drawn. This is the default behavior of PROC GMAP: it draws only map areas that correspond to map areas found in the response data set.

Before you can draw a map using a map data set with unprojected coordinates, the coordinates must be projected using PROC GPROJECT.

Example 1.12 Create a choropleth map drawn with projected STATES map data set

```
proc format;
value pop
low     -< 1300000 = '<1.3'
1300000 -< 4000000 = '1.3-3.9'
4000000 -< 6000000 = '4.0-5.9'
6000000 -  high    = '6.0+'
;
run;

proc gproject
data=maps.states ❶
out=projected_states ❷
;
id state; ❸
run;

pattern1 v=ms c=grayfa; pattern2 v=ms c=grayda;
pattern3 v=ms c=grayaa; pattern4 v=ms c=gray5a;

title 'YEAR 2000 CENSUS POPULATION';

proc gmap
map=projected_states ❹
data=us2000st
;
id state;
choro pop2000 / discrete coutline=black;
label pop2000 = 'MILLIONS';
format pop2000 pop.;
run;
quit;
```

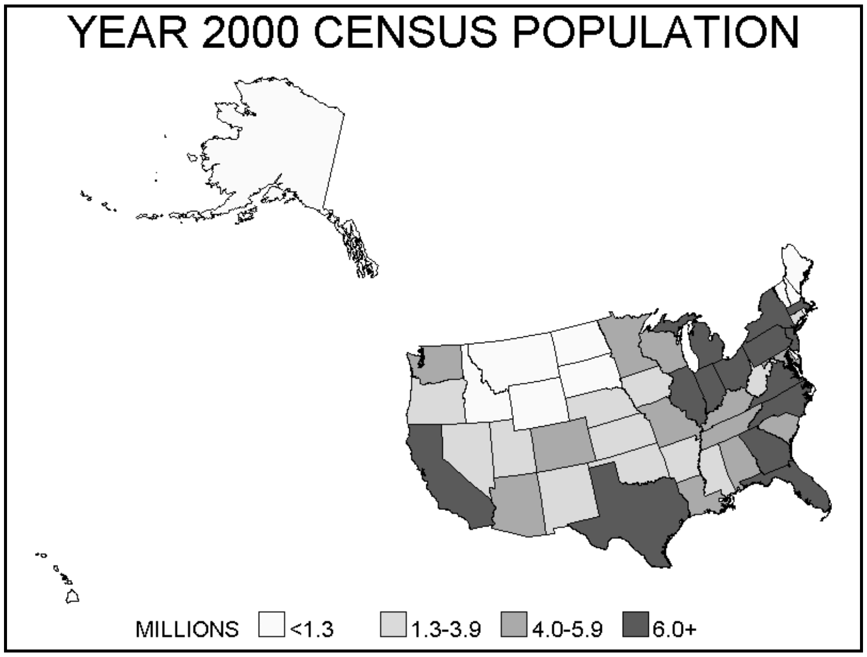

Figure 1.12 *Map drawn using projected STATES map data set*

PROC GPROJECT requires as input a map data set containing unprojected X-Y coordinates; in this example STATES is used. ❶ It creates a new map data set with projected coordinates. ❷ In addition to variables X and Y, the input data set must also contain a variable that associates each X-Y coordinate with a map area. This variable (STATE in the STATES data set) is used in the ID statement. ❸ The projected map data set is used in PROC GMAP ❹ to produce the map shown in Figure 1.12.

By default PROC GPROJECT expects the input data set to have the X-Y coordinates in radians and the X values to increase in value from east to west. All map data sets supplied by SAS that contain unprojected X-Y coordinates have these two features. Two options available in PROC GPROJECT can be used if a user-supplied map data set does not have either or both features: DEGREE, if coordinates are in degrees rather than radians; EASTLONG, if X values increase from west to east. A number of other options available in PROC GPROJECT are discussed in a later chapter.

You can exclude any areas that will not be displayed in the map by using a WHERE statement in PROC GPROJECT prior to using PROC GMAP.

Example 1.13 Create a projected STATES map excluding several areas

```
proc format;
value pop
low     -< 1300000 = '<1.3'
1300000 -< 4000000 = '1.3-3.9'
4000000 -< 6000000 = '4.0-5.9'
6000000 -  high    = '6.0+'
;
run;

proc gproject
data=maps.states
out=projected_states
;
where fipstate(state) not in ('AK','HI','PR'); ❶
id state;
run;

pattern1 v=ms c=grayfa;
pattern2 v=ms c=grayda;
pattern3 v=ms c=grayaa;
pattern4 v=ms c=gray5a;

title 'YEAR 2000 CENSUS POPULATION';

proc gmap
map=projected_states
data=us2000st
;
id state;
choro pop2000 / discrete coutline=black;
label pop2000 = 'MILLIONS';
format pop2000 pop.;
run;
quit;
```

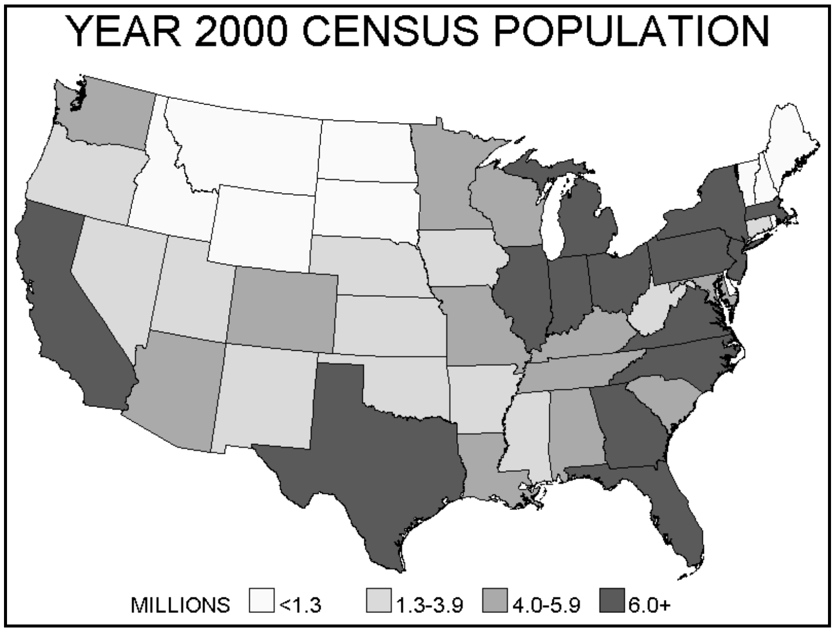

Figure 1.13 *Projected STATES map set (Alaska, Hawaii, and Puerto Rico excluded in PROC GPROJECT)*

This example uses a WHERE statement in PROC GPROJECT to exclude three map areas. ❶ The FIPSTATE function is used (as described after Example 1.10), allowing the states to be specified with two-character postal codes. AK, HI, and PR are the codes for Alaska, Hawaii, and Puerto Rico, respectively.

Map data sets supplied by SAS use two different terms to refer to unprojected X-Y coordinates, unprojected and deprojected. Although there are differences in the origins of the observations referred to by the different terms, map data sets with unprojected or deprojected observations should be treated the same—that is, the deprojected map data sets must also be projected with PROC GPROJECT prior to being used in PROC GMAP.

1.4.3 PROC GREDUCE

Several of the differences between the US and STATES map data sets have already been discussed, with emphasis on the difference between projected and unprojected X-Y coordinates. The two map data sets differ in two more respects. The STATES data set contains an additional variable, density, and it has over 50,000 observations while the US data set has just over 1,500. The large number of coordinates adds detail to maps drawn with the STATES map data set. Even on small maps, this enables you to see the extra detail along the coastlines and in the boundaries between the map areas. However, maps drawn with the US map data set, with only 1,500+ observations, look fine except for the omission of Long Island from New York State.

You can use PROC GREDUCE to add a density variable to map data sets. The density variable in the STATES map data set contains information about how much detail a given observation (X-Y coordinate) contributes to a map drawn using the data set. The procedure assesses how important each X-Y coordinate is in maintaining the original shape of map areas. The density variable can be used to reduce the complexity of map boundaries by eliminating those observations containing X-Y coordinates that are nonessential—that is, observations whose elimination does not significantly alter the appearance of the original map.

You can use PROC FREQ to create a table showing the number of observations with various levels of the variable density in the STATES data set.

DENSITY FOR LOWER-RESOLUTION MAPS

DENSITY	FREQUENCY	PERCENT	CUMULATIVE FREQUENCY	CUMULATIVE PERCENT
0	1268	2.41	1268	2.41
1	1901	3.61	3169	6.02
2	1969	3.74	5138	9.75
3	10250	19.46	15388	29.21
4	37294	70.79	52682	100.00

The higher the value of the density variable, the more detail a given point adds to map area boundaries and the less important that point is in maintaining the overall appearance of the original map.

Example 1.14 Create a projected states map excluding observations with a density of 3 or greater

```
proc format;
value pop
low     -< 1300000 = '<1.3'
1300000 -< 4000000 = '1.3-3.9'
4000000 -< 6000000 = '4.0-5.9'
6000000 -  high    = '6.0+'
;
run;

proc gproject
data=maps.states
out=projected_states
;
where fipstate(state) not in ('AK','HI','PR') and density le 2; ❶
id state;
run;
```

```
pattern1 v=ms c=grayfa;
pattern2 v=ms c=grayda;
pattern3 v=ms c=grayaa;
pattern4 v=ms c=gray5a;

title 'YEAR 2000 CENSUS POPULATION';

proc gmap
map=projected_states
data=us2000st
;
id state;
choro pop2000 / discrete coutline=black;
label pop2000 = 'MILLIONS';
format pop2000 pop.;
run;
quit;
```

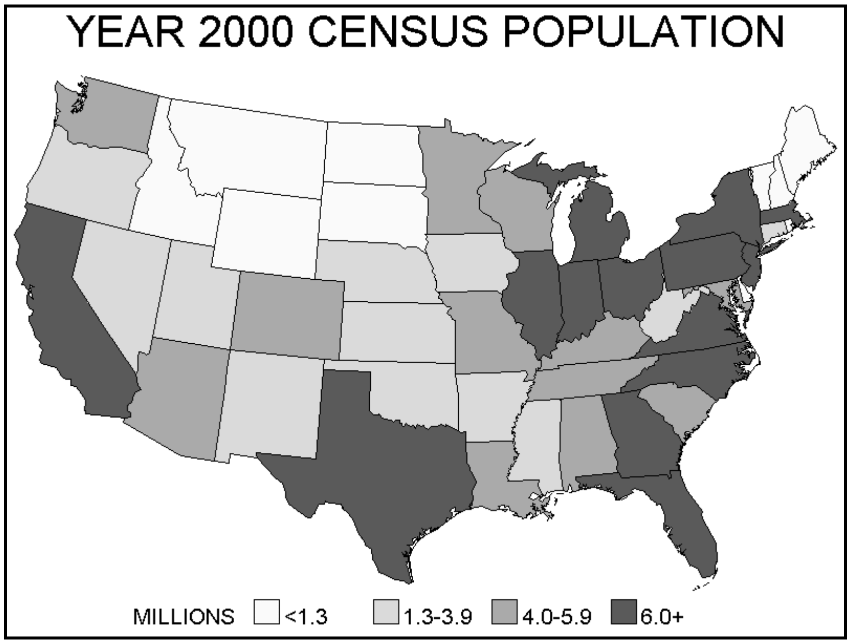

Figure 1.14 *Map created with limited observations from STATES map data set*

This example uses the WHERE statement in PROC GPROJECT to exclude three map areas (Alaska, Hawaii, and Puerto Rico), plus any observation with a density of three or greater.❶ Rather than using the FIPS codes for each state, this example uses the FIPSTATE function and postal codes (as was shown as an alternative method in Example 1.13). The map in Figure 1.14 looks similar to the map in Figure 1.13, though it was created using approximately

5,100 X-Y coordinates compared with the approximately 50,000 used for Figure 1.13. Using fewer observations allows maps to be drawn more quickly. If the output of PROC GMAP is being directed to a file rather than to the graph output window, maps drawn with fewer points require less storage space.

Not every map data set supplied by SAS contains a density variable. It is also unlikely that a user-supplied map data set would contain such a variable. PROC GREDUCE can be used to add a density variable to a map data set. This procedure requires a map data set as input and the specification of the variable within the data set that identifies map areas. PROC GREDUCE produces a new map data set that contains a density variable.

Example 1.15 Add a density variable to a map data set

```
proc greduce data=maps.mexico out=mexico_dens; ❶
id id; ❷
run;

proc freq data=mexico_dens; ❸
table density;
run;

pattern v=me c=black r=32; ❹
title h=6 "MAP CREATED WITH MEXICO MAP DATA SET";

proc gmap
map=mexico_dens ❺
data=mexico_dens
;
id id;
choro id / discrete nolegend;
where density le 4; ❻
run;
quit;
```

DENSITY	FREQUENCY	PERCENT	CUMULATIVE FREQUENCY	CUMULATIVE PERCENT
0	166	3.64	166	3.64
1	4	0.69	170	3.73
2	276	6.05	446	9.78
3	553	12.12	999	21.90
4	895	19.62	1894	41.53
5	952	20.87	2846	62.40
6	1715	37.60	4561	100.00

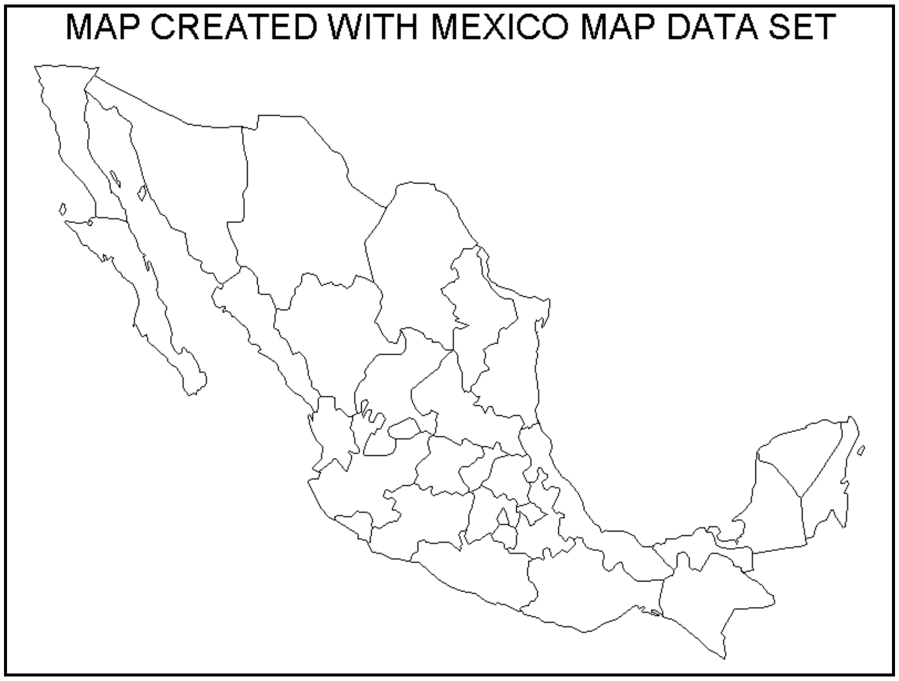

Figure 1.15 *Map created with limited observations after creating a density variable with PROC GREDUCE*

This example uses PROC GREDUCE to add a density variable to the MEXICO map data set, ❶ one of the map data sets supplied by SAS. The variable in the data set that identifies map areas is ID. ❷ A table of density values is created with PROC FREQ. ❸ The new variable can be used to reduce the number of observations used to create a map of Mexico. Thirty-two empty patterns are created, one for each map area in the Mexico map data set. ❹ The R= option specifies that the specified pattern be repeated 32 times. The new map data set, produced by PROC GREDUCE, is used to draw a map of Mexico, ❺ and a WHERE statement limits the observations used to draw the map to only those with a density of 4 or less ❻ (less than half the observations in the map data set).

Numerous options can be used in PROC GREDUCE. You can specify the maximum number of observations at various levels of the variable density using options N1 though N5 (the maximum number of points with a density of 1 through 5). Another set of options (E1 through E5) allows you to control the assignment of observations to various levels of density using a distance measure. The map in Figure 1.15 illustrates that using the default values of the various options can produce an acceptable map.