CHAPTER

19

The GMAP Procedure

Overview

The GMAP procedure produces two-dimensional (choropleth) or three-dimensional (block, prism, and surface) color maps that show variations of a variable value with respect to an area. A wide assortment of geographic maps are available with SAS/GRAPH software, and you also can create your own geographic or spatial maps.

Use the GMAP procedure to

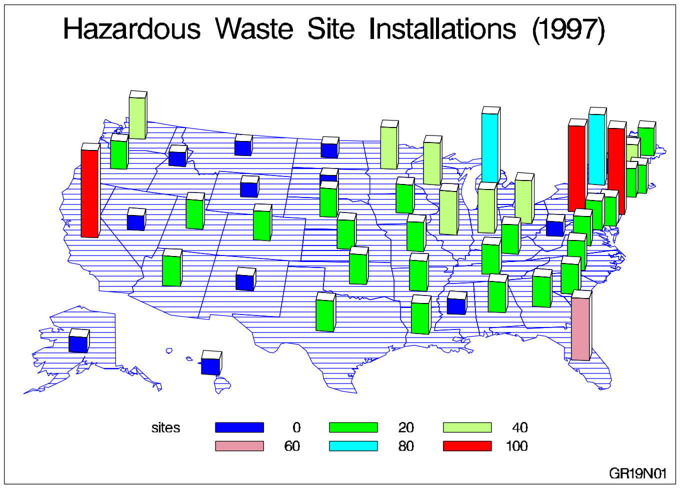
- summarize data that vary by physical area
- show trends and variations of data between geographic areas
- highlight regional differences or extremes
- produce maps.

About Block Maps

Block maps display a block at the approximate center of each map area to convey information about response variable values. The height of each block represents a response level. The height of the blocks is not directly proportional to the value of the response variable. Instead, the block heights increase in order of the response levels.

Figure 19.1 on page 732 shows a simple block map of hazardous waste sites that are installed in each state. Each state is a midpoint. The number of sites in each state (the response value) is represented by the height of the block.

Figure 19.1 Block Map (GR19N01)



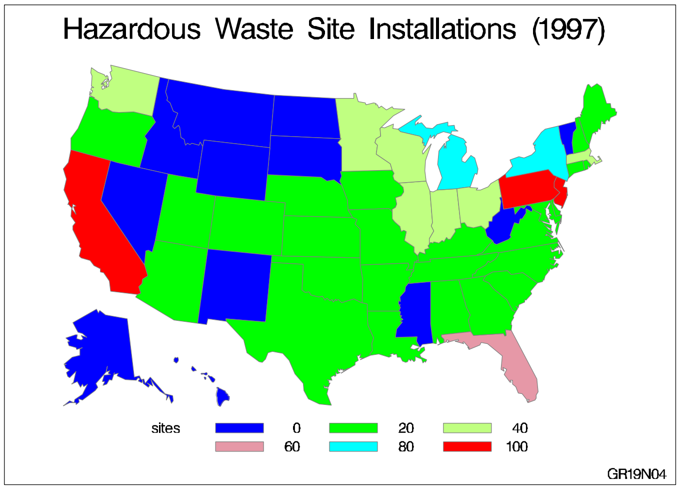
The program for this map is in Example 1 on page 771. For more information on producing block maps, see “BLOCK Statement” on page 742.

About Choropleth Maps

Two-dimensional (choropleth) maps indicate levels of magnitude or response levels of the corresponding response variable by filling map areas with different colors and patterns.

Figure 19.2 on page 733 shows a choropleth map of hazardous waste sites that are installed in each state. Each state is a midpoint. The number of sites in each state (the response value) is represented by the pattern that is assigned to the state.

Figure 19.2 Two-dimensional (Choropleth) Map (GR19N04)



The program for this map is in Example 4 on page 778.

You can also produce a simple choropleth map that shows an outline of a map’s areas by specifying your map data set as both the map data set and the response data set in a GMAP statement. For more information on producing choropleth maps, see “CHORO Statement” on page 748.

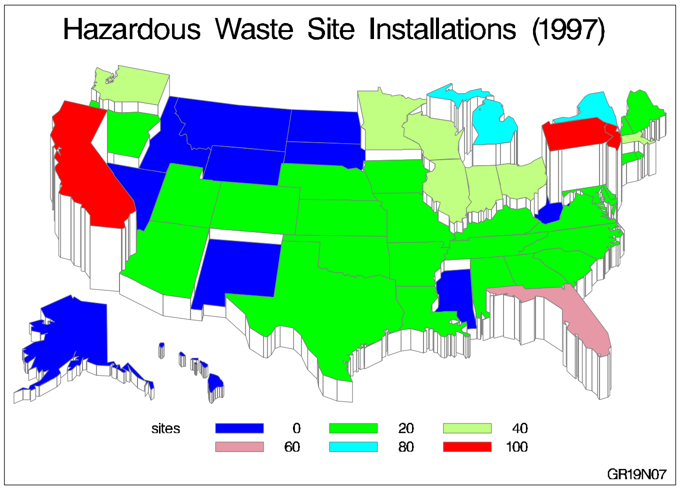
About Prism Maps

Prism maps use polyhedrons (raised polygons) in the shape of each map area to convey information about response variable values. The height of each polyhedron, or prism, represents an ordinal level of the response variable. Prism heights increase in order of response levels. That is, the lowest prisms correspond to the first level, and the tallest prisms correspond to the last level.

You can alter the perspective of the map by selecting a viewing position (the point in space from which you view the map). You can also change the position of the light source so that the shadowing on the prisms enhances the illusion of height.

Figure 19.3 on page 734 shows a prism map of hazardous waste sites installed in each state. Each state is a midpoint. The number of sites in each state (the response value) is represented by the height of the state.

Figure 19.3 Prism Map (GR19N07)



The program for this map is in Example 7 on page 788. For more information on producing prism maps, see "PRISM Statement" on page 753.

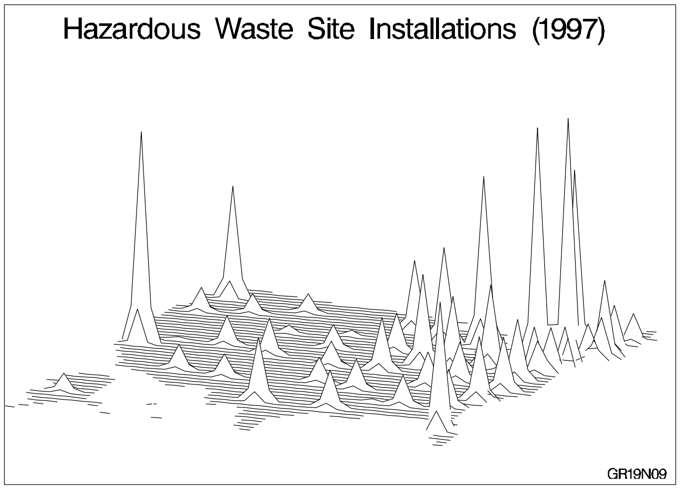
About Surface Maps

Surface maps display a spike at the approximate center of each map area to convey information about response variable values. The height of the spike corresponds to the relative value of the response variable, not to the actual value of the response variable. Thus, a spike that represents a value of 100 may not be exactly 10 times higher than a spike that represents a value of 10. Map area boundaries are not drawn.

Surface maps provide no clear map area boundaries and no legend. Thus, surface maps provide a simple way to judge relative trends in the response data but are an inappropriate way to represent specific response values.

Figure 19.4 on page 734 shows a surface map of hazardous waste sites that are installed in each state. Each state is a midpoint. The number of sites in each state (the response value) is represented by the height of the spike.

Figure 19.4 Surface Map (GR19N09)



The program for this map is in Example 9 on page 791. For more information on producing surface maps, see “SURFACE Statement” on page 759.

Concepts

The GMAP procedure requires a map data set and a response data set. These two data sets must contain the required variables or the procedure stops with an error message. You can use the same data set as both the map data set and the response data set, as long as the requirements are met. If a different data set is used as the response data set, it must contain an ID variable that is identical to the ID variable in the map data set.

About Map Data Sets

A *map data set* is a SAS data set that contains coordinates that define the boundaries of map areas, such as states or counties. A map data set must contain at least these variables:

- a numeric variable named X that contains the horizontal coordinates of the boundary points. The value of this variable could be either projected or unprojected. If unprojected, X represents longitude.
- a numeric variable named Y that contains the vertical coordinates of the boundary points. The value of this variable could be either projected or unprojected. If unprojected, Y represents latitude.
- one or more variables that uniquely identify the areas in the map. Map area identification variables can be either character or numeric and are indicated in the ID statement.

The X and Y variable values in the map data set do not have to be in any specific units because they are rescaled by the GMAP procedure based on the minimum and maximum values in the data set. The minimum X and Y values are in the lower-left corner of the map, and the maximum X and Y values are in the upper-right corner.

Map data sets in which the X and Y variables contain longitude and latitude should be projected before you use them with PROC GMAP. See Chapter 23, “The GPROJECT Procedure,” on page 873 for details.

Optionally, the map data set also can contain a variable named SEGMENT to identify map areas that comprise noncontiguous polygons. Each unique value of the SEGMENT variable within a single map area defines a distinct polygon. If the SEGMENT variable is not present, each map area is drawn as a separate closed polygon that indicates a single segment.

The observations for each segment of a map area in the map data set must occur in the order in which the points are to be joined. The GMAP procedure forms map area outlines by connecting the boundary points of each segment in the order in which they appear in the data set, eventually joining the last point to the first point to complete the polygon.

Any variables in the map data set other than the ones mentioned above are ignored for the purpose of determining map boundaries.

About SAS/GRAPH Map Data Sets

In addition to the variables described in “About Map Data Sets” on page 735, the SAS/GRAPH map data sets may also contain the following variables:

- a numeric variable named LONG containing the unprojected longitude in radians of the boundary points.
- a numeric variable named LAT containing the unprojected latitude in radians of the boundary points.

The GMAP procedure uses the values of the X and Y variables to draw the map. Therefore, if you want to produce an unprojected map by using the values in LONG and LAT, you would have to rename LONG and LAT to X and Y first.

SAS/GRAPH includes a number of predefined map data sets. These data sets are described in “SAS/GRAPH Map Data Sets” on page 761.

Map Data Sets Containing X, Y, LONG, and LAT

Most Institute-supplied map data sets contain four coordinate variables (X, Y, LONG, and LAT). In this case, X and Y are always projected values that will be used by the GRAPH procedures (by default). If you need to use the unprojected values that are contained in the LONG and LAT variables, you will need to rename the LONG and LAT variables to X and Y since the GMAP procedure automatically uses X and Y. See “Input Map Data Sets that Contain Both Projected and Unprojected Values” on page 875 for more details.

Map Data Sets Containing Only X and Y

The Institute-supplied map data sets that contain X and Y variables (and no LONG and LAT variables), are usually projected maps. However, there are a few map data sets for the US and Canada that contain X and Y values that are unprojected longitude and latitude. In this case, you will need to use the GPROJECT procedure to project the map (see Chapter 23, “The GPROJECT Procedure,” on page 873).

Note: You can determine whether a SAS map data set is projected or unprojected by looking at the description of each variable that is displayed when you use the CONTENTS procedure or by browsing the MAPS.METAMAPS data set. △

Specialty Map Data Sets

There are several map data sets available with SAS/GRAPH that allow you to easily label maps:

MAPS.USCENTER
: contains the X and Y coordinates of the visual center of each state in the U.S. and Washington, D.C., as well as points in the ocean for states that are too small to contain a label. You can use MAPS.USCENTER with the MAPS.US, MAPS.USCOUNTY, MAPS.COUNTIES, and MAPS.COUNTY data sets.

MAPS.USCITY
: contains the X and Y coordinates of selected cities in the U.S. Many city names occur in more than one state, so you may have to subset by state to avoid duplication. You can use MAPS.USCITY with the MAPS.US, MAPS.USCOUNTY, MAPS.COUNTIES, and MAPS.COUNTY data sets.

MAPS.CANCENS
: contains the names of the Canadian census divisions. You can use MAPS.CANCENS with the MAPS.CANADA and MAPS.CANADA3 data sets.

See the MAPS.METAMAPS data set for details on each of the Institute-supplied map data sets.

About Response Data Sets

A *response data set* is a SAS data set that contains

- □ one or more response variables that contain data values that are associated with map areas. Each value of the response variable is associated with a map area in the map data set.
- □ identification variables that identify the map area to which a response value belongs. These variables must be the same as those that are contained in the map data set.

The response data set can contain other variables in addition to these required variables.

The values of the map area identification variables in the response data set determine the map areas to be included on the map unless you use the ALL option in the PROC GMAP statement. That is, unless you use ALL in the PROC GMAP statement, only the map areas with response values are shown on the map. As a result, you do not need to subset your map data set if you are mapping only a small section of the map. However, if you map the same small section frequently, create a subset of the map data set for efficiency.

For choropleth, block, and prism maps, the response variables can be either character or numeric. For surface maps, the response variables must be numeric with only positive values.

About Response Variables

The GMAP procedure can produce block, choropleth, and prism maps for both numeric and character response variables. Numeric variables fall into two categories: discrete and continuous.

- □ *Discrete variables* contain a finite number of specific numeric values that are to be represented on the map. For example, a variable that contains only the values 1989 or 1990 is a discrete variable.
- □ *Continuous variables* contain a range of numeric values that are to be represented on the map. For example, a variable that contains any real value between 0 and 100 is a continuous variable.

Numeric response variables are always treated as continuous variables unless the DISCRETE option is used in the action statement.

About Response Levels

Response levels are the values that identify categories of data on the graph. The categories that are shown on the graph are based on the values of the response variable. Based on the type of the response variable, a response level can represent these values:

- □ a specific character value. If the response variable is character type, the GMAP procedure treats each unique value of the variable as a response level. For example, if the response variable contains the names of ten regions, each region will be a response level, resulting in ten response levels.

  The exception to this is that the MIDPOINTS= option chooses specific response level values. Any response variable values that do not match one of the specified response level values are ignored. For example, if the response variable contains the names of ten regions and you specify these midpoints, only the observations for **Midwest**, **Northeast**, and **Northwest** are included on the map:

  ```
  midpoints='Midwest' 'Northeast' 'Northwest'
  ```

- a range of numeric values. If the response variable is numeric, the GMAP procedure determines the number of response levels for the response variable. Each response level then represents the median of a range of values.

  These options are exceptions to this:

  - The LEVELS= option specifies the number of response levels to be used on the map.
  - The DISCRETE option causes the numeric variable to be treated as a discrete variable.
  - The MIDPOINTS= option chooses specific response level values as medians of the value ranges.

  If the response variable values are continuous, the GMAP procedure assigns response level intervals automatically unless you specify otherwise. The response levels represent a range of values rather than a single value.

- a specific numeric value. If the response variable is numeric and you use the DISCRETE option, the GMAP procedure treats the variable much the same way as it treats a character response variable. That is, the procedure creates a response level for each unique value of the response variable. If you use DISCRETE with a numeric response variable that has an associated format, each formatted value is represented by a different response level. Formatted values are truncated to 16 characters.

The BLOCK, CHORO, and PRISM statements assign patterns to response levels. In CHORO and PRISM maps, response levels are shown as map areas. However, in BLOCK maps, response levels are shown as blocks. The default fill pattern for the response level is solid.

PATTERN statements can define the fill patterns and colors for both blocks and map areas. PATTERN definitions that define valid block patterns are applied to the blocks (response levels), and PATTERN definitions that define valid map patterns are applied to map areas.

See "PATTERN Statement" on page 211 for more information on fill pattern values and default pattern rotation.

About Identification Variables

Identification (ID) variables are common to both the map data set and the response data set. They identify the map areas (for example, counties, states, or provinces) that make up the map. A *unit area* or *map area* is a group of observations with the same ID value. The GMAP procedure matches the value of the response variables for each map area in the response data set to the corresponding map area in the map data set to create the output graphs.

Displaying Map Areas and Response Data

Whether the GMAP procedure draws a map area and whether it displays patterns for response values depends on the contents of the response data set and on the ALL and MISSING options. describes the conditions under which the procedure does or does not display map areas and response data.

If the response data set...	*And if...*	*Then the procedure...*
includes the map area	the map area has a response value	draws the map area and displays the response data
includes the map area	the map area has no response value (that is, the value is missing)	draws the map area but leaves it empty
includes the map area	the map area has no response value and the MISSING option is used in the map statement	draws the map area and displays a response level for the missing value
does not include the map area	the ALL option is used in the PROC GMAP statement	draws the map area but leaves it empty
does not include the map area	the ALL option is not used	does not draw the map area

Summary of Use

To use the GMAP procedure, you must do the following:

1. If necessary, issue a LIBNAME statement for the SAS data library that contains the map data set that you want to display.
2. Determine what processing needs to be done to the map data set before it is displayed. Use the GPROJECT, GREDUCE, and GREMOVE procedures or a DATA step to perform the necessary processing.
3. Issue a LIBNAME statement for the SAS data set that contains the response data set, or use a DATA step to create a response data set.
4. Use the PROC GMAP statement to identify the map and response data sets.
5. Use the ID statement to name the identification variable(s).
6. Use a BLOCK, CHORO, PRISM, or SURFACE statement to identify the response variable and generate the map.

Procedure Syntax

Requirements: One ID statement and at least one CHORO, BLOCK, PRISM, or SURFACE statement are required.

Global statements: FOOTNOTE, LEGEND, PATTERN, TITLE

Reminder: The procedure can include the BY, FORMAT, LABEL, and WHERE statements as well as the SAS/GRAPH NOTE statement.

Supports: RUN-group processing Output Delivery System (ODS)

PROC GMAP MAP=*map-data-set*
<DATA=*response-data-set*>
<ALL>
<ANNOTATE=*Annotate-data-set*>
<GOUT=<*libref.*>*output-catalog*>
<IMAGEMAP=*output-data-set*>;

ID*id-variable(s)*;

BLOCK *response-variable(s)* </ *option(s)*>;
CHORO *response-variable(s)* </ *option(s)*>;

PRISM *response-variable(s)</ option(s)>*; **SURFACE** *response-variable(s) </ option(s)>*;

PROC GMAP Statement

Identifies the map data set and the response data set that contains the variables associated with the map. Optionally causes the procedure to display all map areas and specifies annotation and an output catalog.

Requirements: Both a map data set and a response data set are required.

PROC GMAP MAP=*map-data-set*
<DATA=*response-data-set*>
<ALL>
<ANNOTATE=*Annotate-data-set*>
<GOUT=<*libref.*>*output-catalog*>
<IMAGEMAP=*output-data-set*>;

Required Arguments

MAP=*map-data-set*
names a SAS map data set that contains the Cartesian coordinates for the boundary points of each map area. The map data set also must contain the same identification variable or variables as the response data set.

See also: "About Map Data Sets" on page 735

Options

PROC GMAP statement options affect all of the graphs that are produced by the procedure.

ALL
specifies that all maps generated by the procedure should include every map area from the map data set, even if the response data set does not include an observation for the map area. When ALL is used, the map areas that are not in the response data set are empty (no pattern fill) and are outlined in the foreground color. To change the outline color of an empty area, use the CEMPTY= option in the MAP statement.

If you omit this option, the GMAP procedure does not draw those map areas in the map data set that have no corresponding observations in the response data set. This is the default behavior.

When you use the ALL option with BY-group processing, the maps that are generated for each BY group include every map area from the map data set.

See also: "Displaying Map Areas and Response Data" on page 738

ANNOTATE=*Annotate-data-set*
ANNO=*Annotate-data-set*
specifies a data set to annotate all of the maps that are produced by the GMAP procedure. To annotate individual maps, use ANNOTATE= in the action statement.

See also: Chapter 10, "The Annotate Data Set," on page 403

DATA=*response-data-set*
identifies the SAS data set that contains the response values that are evaluated and represented on the map. By default, the procedure uses the most recently created SAS data set.

See also: "About Response Data Sets" on page 737 and "SAS Data Sets" on page 25

GOUT=<*libref.*>*output-catalog*
specifies the SAS catalog in which to save the graphics output that is produced by the GMAP procedure. If you omit the libref, SAS/GRAPH looks for the catalog in the temporary library called WORK and creates the catalog if it does not exist.

See also: "Storing Graphics Output in SAS Catalogs" on page 49

IMAGEMAP=*output-data-set*
creates a SAS data set that contains information about the graph and about areas in the graph. This information includes the shape and coordinates of the areas, and is used to build an HTML file that links the graph areas to other files or images. This linking provides drill-down functionality on the graph. The Imagemap data set also contains the information that is stored in the variables referenced by the HTML= and HTML_LEGEND= options. Therefore, in order to use IMAGEMAP= to create an HTML file, you must also use the HTML= option or the HTML_LEGEND= option or both.

See also: "Customizing Web Pages for Drill-down Graphs" on page 100

ID Statement

Identifies the variable or variables in both the map data set and the response data set that define map areas.

Requirements: At least one *id-variable* is required.

ID *id-variable(s)*;

Required Arguments

id-variable(s)
identifies one or more variables in the input data sets that define map areas. Every variable that is listed in the ID statement must appear in both the map and response data sets. *Id-variable* can be either numeric or character and should have the same name, type, and length in both the response and map data sets.

See also: "About Identification Variables" on page 738

Featured in: Example 1 on page 771, Example 3 on page 775 and Example 4 on page 778

BLOCK Statement

Creates three-dimensional block maps on which levels of magnitude of the specified response variables are represented by blocks of varying height, pattern, and color.

Requirements: At least one response variable is required. The ID statement must be used in conjunction with the BLOCK statement.

Global statements: FOOTNOTE, LEGEND, PATTERN, TITLE

Description

The BLOCK statement specifies the variable or variables that contain the data that are represented on the map by blocks of varying height, pattern, and color. This statement automatically

- □ determines the midpoints
- □ scales the blocks
- □ assigns patterns to the block faces and map areas. (See “About Block Maps and Patterns” on page 748 for more information.)

You can use statement options to enhance the appearance of the map. For example, you can specify the width of the blocks, the outline colors for the blocks and the map areas, and the angle of view. Other statement options control the response levels.

In addition, you can use global statements to modify the block patterns, the map patterns, and the legend, as well as to add titles and footnotes to the map. You can also use an Annotate data set to enhance the map.

BLOCK*response-variable(s) </ option(s)>*;

option(s) can be one or more options from any or all of the following categories:

- □ appearance options:

  ANNOTATE=*Annotate-data-set*

  BLOCKSIZE=*size*

  CBLKOUT=*block-outline-color* | SAME

  CEMPTY=*empty-area-outline-color*

  COUTLINE=*nonempty-area-outline-color* | SAME

  SHAPE=*3D-block-shape*

  XSIZE=*map-width <units>*

  YSIZE=*map-height <units>*

  XVIEW=*x*

  YVIEW=*y*

  ZVIEW=*z*

- □ mapping options:

  AREA=*n*

  DISCRETE

  LEVELS=*number-of-response-levels*

  MIDPOINTS=*value-list*

  MISSING

- □ legend options:

CTEXT=*text-color*

LEGEND=LEGEND<1...99>

NOLEGEND

- description options:

  DESCRIPTION='*entry-description*'

  NAME='*entry-name*'

- ODS options

  HTML=*variable*

  HTML_LEGEND=*variable*

Required Arguments

response-variable(s)

specifies one or more variables in the response data set that contains response values that are represented on the map. Each response variable produces a separate map. All variables must be in the input data set. Separate multiple response variables with blanks.

Blocks are not drawn for missing values for the response variable unless you use the MISSING option in the BLOCK statement.

See also: "About Response Variables" on page 737

Options

Options in a BLOCK statement affect all of the maps that are produced by that statement. You can specify as many options as you want and list them in any order.

ANNOTATE=*Annotate-data-set*
ANNO=*Annotate-data-set*

specifies a data set to annotate maps that are produced by the BLOCK statement.

Note: Annotate coordinate systems 1, 2, 7, and 8 are not valid with block maps. △

See also: Chapter 10, "The Annotate Data Set," on page 403

AREA=*n*

specifies that a different map pattern be used for the surface of each map area or group of map areas on the map. The value of *n* indicates which variable in the ID statement determines the groups that are distinguished by a surface pattern. If your ID statement has only one map area identification variable, use AREA=1 to indicate that each map area surface uses a different pattern. If you have more than one variable in your ID statement, use *n* to indicate the position of the variable that defines groups that will share a pattern. When you use AREA=, the map data set should be sorted in order of the variables in the ID statement.

By default, AREA= fills map areas by rotating the default hatch patterns through the colors list, beginning with the M2N0 pattern. Unless specified otherwise, the outline color is the first color in the colors list. If the V6COMP graphics option or a PATTERN statement is specified, then the value of COUTLINE= defaults to SAME.

You can specify pattern fills and/or colors with PATTERN statements that specify map/plot patterns. A separate PATTERN definition is needed for each specified area. For more information about default pattern behavior or pattern specifications, see "PATTERN Statement" on page 211.

Featured in: Example 3 on page 775

BLOCKSIZE=*size*

specifies the width of the blocks. The unit for *size* is the character cell width for the selected output device. By default, BLOCKSIZE=2.

Featured in: Example 5 on page 779

CBLKOUT=*block-outline-color* | SAME

outlines all blocks in the specified color. SAME specifies that the outline color of a block or a block segment or a legend value is the same as the interior pattern color.

The default outline color depends on the PATTERN statement:

- □ If no PATTERN statements are specified, the default outline color is the foreground color (the first color in the colors list).
- □ If a PATTERN statement or the V6COMP graphics option is specified, the default is CBLKOUT=SAME.

CBLKOUT= is not valid when SHAPE=CYLINDER.

Note: If you specify empty block patterns, (VALUE=EMPTY in a PATTERN statement) you should not change the outline color from the default value, SAME, to a single color. Otherwise all the outlines will be one color and you will not be able to distinguish between the empty areas. △

Featured in: Example 1 on page 771 and Example 3 on page 775

CEMPTY=*empty-area-outline-color*

outlines empty map areas in the specified color. This option affects only map areas that are empty. Empty map areas are generated in block maps only when a map area is omitted from the response data set and the ALL option is included in the PROC GMAP statement.

The default outline color is the same as the default COUTLINE= color.

See also: ALL on page 740 and “Displaying Map Areas and Response Data” on page 738

COUTLINE=*nonempty-area-outline-color* | SAME

outlines non-empty map areas in the specified color. SAME specifies that the outline color of a map area is the same as the interior pattern color.

The default outline color depends on the PATTERN statement:

- □ If no PATTERN statement is specified, the default outline color is the foreground color (the first color in the colors list).
- □ If a PATTERN statement or the V6COMP graphics option is specified, the default is COUTLINE=SAME.

Note: If you specify empty map patterns, (VALUE=MEMPTY in a PATTERN statement) you should not change the outline color from the default value, SAME, to a single color. Otherwise all the outlines will be one color and you will not be able to distinguish between the empty areas. △

Featured in: Example 3 on page 775

CTEXT=*text-color*

specifies a color for the text in the legend. If you omit the CTEXT= option, a color specification is searched for in this order:

1. the CTEXT= option in a GOPTIONS statement
2. the default, the first color in the colors list.

The CTEXT= color specification is overridden if you also use the COLOR= suboption of a LABEL= or VALUE= option in a LEGEND definition that is assigned to the map legend. The COLOR= suboption determines the color of the legend label or the color of the legend value descriptions, respectively.

DESCRIPTION='*entry-description*'
DES='*entry-description*'
specifies the description of the catalog entry for the map. The maximum length for *entry-description* is 40 characters. The description does not appear on the map. By default, the GMAP procedure assigns a description of the form BLOCK MAP OF *variable*, where *variable* is the name of the map variable.

Featured in: Example 5 on page 779

DISCRETE
treats a numeric response variable as a discrete variable rather than as a continuous variable. When you use DISCRETE, the response variable values are not grouped into ranges; instead, the GMAP procedure uses a separate response level (block height, pattern, and color) for each different value of the formatted response variable. The LEVELS= option is ignored when you use the DISCRETE option.

Use this option if your numeric response variable is assigned a user-written format.

Note: If the data do not contain a value in a particular range of the format, that formatted range is not displayed in the legend. △

Featured in: Example 3 on page 775 and Example 5 on page 779 (with the CHORO statement)

HTML=*variable*
identifies the variable in the input data set whose values create links in the HTML file created by the ODS HTML statement. These links are associated with an area of the chart and point to the data or graph you wish to display when the user drills down on the area.

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links in the HTML file created by the ODS HTML statement. These links are associated with a legend value and point to the data or graph you wish to display when the user drills down on the value.

LEGEND=LEGEND<1...99>
assigns the specified LEGEND definition to the map legend. LEGEND= is ignored if the specified LEGEND definition is not currently in effect. In the GMAP procedure, the BLOCK statement produces a legend unless you use the NOLEGEND option. If you use the SHAPE= option in a LEGEND statement, only the value BAR is valid.

See also: "LEGEND Statement" on page 187

Featured in: Example 2 on page 773 and Example 5 on page 779

LEVELS=*number-of-response-levels*
specifies the number of response levels that are to be graphed when the response variables are continuous. Each level is assigned a different block height, pattern, and color combination.

If you do not use the LEVELS= option or the DISCRETE option, the GMAP procedure determines the number of response levels that use the formula FLOOR(1+3.3 log(N)), where N is the number of unique map area identification variable values.

The LEVELS= option is ignored when you use the DISCRETE option.

Featured in: Example 2 on page 773

MIDPOINTS=*value-list*

specifies the response levels for the range of response values that are represented by each level (block height, pattern, and color combination).

For numeric response variables, *value-list* is either an explicit list of values or a starting and an ending value with an interval increment, or a combination of both forms:

n <...*n*>

n TO *n* <BY *increment*>

n <...*n* > TO *n* <BY *increment*> <*n*<...*n* >>

By default the increment value is 1. You can specify discrete numeric values in any order. In all forms, *n* can be separated by blanks or commas. For example,

```
midpoints=(2 4 6)
midpoints=(2,4,6)
midpoints=(2 to 10 by 2)
```

If a numeric variable has an associated format, the specified values must be the *unformatted* values. *For character response variables*, *value-list* is a list of unique character values enclosed in quotes and separated by blanks:

'value-1' <...*'value-n'*>

The values are character strings that are enclosed in single quotation marks and separated by blanks. For example,

```
midpoints='Midwest' 'Northeast' 'Northwest'
```

Specify the values in any order. If a character variable has an associated format, the specified values must be the *formatted* values.

You can selectively exclude some response variable values from the map, as shown here:

```
midpoints='Midwest'
```

Only those observations for which the response variable exactly matches one of the values listed in the MIDPOINTS= option are shown on the map. As a result, observations may be excluded inadvertently if values in the list are misspelled or if the case does not match exactly.

Featured in: Example 5 on page 779

MISSING

accepts a missing value as a valid level for the response variable.

See also: “Displaying Map Areas and Response Data” on page 738

NAME='*entry-name*'

specifies the name of the catalog entry for the map. The maximum length for *entry-name* is 8 characters. The default name is GMAP. If the specified name duplicates the name of an existing entry, SAS/GRAPH software adds a number to the duplicate name to create a unique name, for example, GMAP1.

Featured in: Example 5 on page 779

NOLEGEND

suppresses the legend.

SHAPE=*3D-block-shape*
specifies the shape of the blocks. Use this option to enhance the look of the block shape, or to specify a different shape. *3D-block-shape* can be one of the following:

- □ BLOCK
- □ CYLINDER
- □ HEXAGON
- □ PRISM
- □ STAR

The CBLKOUT= option is not valid when SHAPE=CYLINDER.

Featured in: Example 2 on page 584 Example 3 on page 586 Example 7 on page 596

XSIZE=*map-width* <*units*>
YSIZE=*map-height* <*units*>
specify the physical dimensions of the map to be drawn, where *n* is the number of units. By default, the map uses the entire procedure output area.

Valid *units* are CM (centimeters), IN (inches), or PCT (percentage of the graphics output area). By default, the unit is character cells (CELLS).

If you specify values for *n* that are greater than the dimensions of the procedure output area, the map is drawn using the default size.

XVIEW=*x*
YVIEW=*y*
ZVIEW=*z*
specify coordinates of the viewing position in the reference coordinate system. In this system, the four corners of the map lie on the X-Y plane at coordinates (0,0,0), (0,1,0), (1,1,0), and (1,0,0). No axes are actually drawn on the maps that are produced by PROC GMAP, but imagine that the maps are drawn in an X-Y plane.

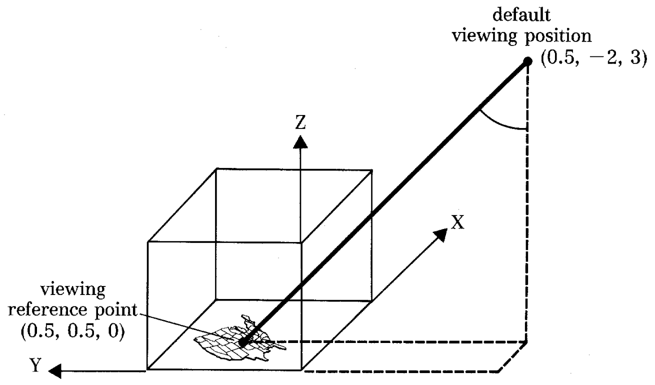
Your viewing position cannot coincide with the viewing reference point at coordinates (0.5,0.5,0), the center of the map. The value for *z* cannot be negative.

If you omit the XVIEW=, YVIEW=, and ZVIEW= options, the default coordinates are (0.5,–2,3). This viewing position is well above and to the south of the center of the map. Specify one, two, or all three of the view coordinates; any that you do not explicitly specify are assigned the default values.

Featured in: Example 2 on page 773

Figure 19.5 on page 747 shows the position of the viewing reference point, as well as the default viewing position.

Figure 19.5 Viewing Position and Viewing Reference Point



About Block Maps and Patterns

Block maps are different from other maps in that they display two different types of areas that use patterns:

- □ the blocks themselves, which represent the response levels
- □ the map areas from which the blocks rise.

By default, the blocks use solid pattern fills and the map areas use a hatch pattern of slanting lines. The map areas in block maps are the *only* map areas that by default do not use solid fills. The map areas and their outlines use the first color in the colors list regardless of whether the list is the device's default colors list or one specified with the COLORS= option in a GOPTIONS statement.

The BLOCK statement has the following options that explicitly control the outline colors used by the blocks and the map areas.

- □ CBLKOUT=
- □ CEMPTY=
- □ COUTLINE=

In addition the AREA= option controls how the map areas are patterned.

When you use PATTERN statements to define the patterns for the map, you must be sure to specify the correct type of pattern for the area. The blocks use bar/block patterns and the map areas use map/plot patterns. See "PATTERN Statement" on page 211 for more information on specifying patterns.

Note: If you specify only one PATTERN statement and include only the COLOR= option, that color will be used for both the blocks and the map areas. For example, this statement makes the blocks solid blue and the map areas blue hatch. △

```
pattern1 color=blue;
```

CHORO Statement

Creates two-dimensional maps in which values of the specified response variables are represented by varying patterns and colors.

Requirements: At least one response variable is required. You must use the ID statement in conjunction with the CHORO statement.

Global statements: FOOTNOTE, LEGEND, PATTERN, TITLE

Description

The CHORO statement specifies the variable or variables that contain the data represented on the map by patterns that fill the map areas. This statement automatically

- □ determines the midpoints
- □ assigns patterns to the map areas.

You can use statement options to enhance the appearance of the map, for example, by selecting the colors and patterns that fill the map areas. Other statement options control the selection of ranges for the response variable.

In addition, you can use global statements to modify the map area patterns and legend, as well as add titles and footnotes to the map. You can also use an Annotate data set to enhance the map.

CHORO*response-variable(s)* </ *option(s)*>;

option(s) can be one or more options from any or all of the following categories:

- appearance options:

  ANNOTATE=*Annotate-data-set*

  CEMPTY=*empty-area-outline-color*

  COUTLINE=*nonempty-area-outline-color* | SAME

  XSIZE=*map-width* <*units*>

  YSIZE=*map-height* <*units*>

- mapping options:

  DISCRETE

  LEVELS=*number-of-response-levels*

  MIDPOINTS=*value-list*

  MISSING

- legend options:

  CTEXT=*text-color*

  LEGEND=LEGEND<1...99>

  NOLEGEND

- description options:

  DESCRIPTION=’*entry-description*’

  NAME=’*entry-name*’

- ODS options

  HTML=*variable*

  HTML_LEGEND=*variable*

Required Arguments

response-variable(s)

specifies one or more variables in the response data set that contains response values that are represented on the map. Each response variable produces a separate map. All variables must be in the input data set. Separate multiple response variables with blanks.

Missing values for the response variable are not considered valid response values unless you use the MISSING option.

Response variables can be either numeric or character. Numeric response variables with continuous values are grouped into ranges, or response levels. Each response level is assigned a different combination of pattern and color. Character variables and numeric variables (when you use the DISCRETE option) have a different response level for each unique response variable value. Numeric variables are treated as continuous unless you use DISCRETE.

For numeric response variables with continuous values, the MIDPOINTS= or LEVELS= option controls the selection of response level ranges.

See also: “About Response Variables” on page 737

Options

Options in a CHORO statement affect all graphs that are produced by that statement. You can specify as many options as you want and list them in any order.

ANNOTATE=*Annotate-data-set*
ANNO=*Annotate-data-set*
specifies a data set to annotate maps that are produced by the CHORO statement.

See also: Chapter 10, "The Annotate Data Set," on page 403

Featured in: Example 6 on page 786

CEMPTY=*empty-area-outline-color*
outlines empty map areas in the specified color. This option affects the map areas that are empty. Empty map areas are generated in choro maps either

- □ when there is no response value for a map area and the MISSING option is not used, or
- □ when a map area is omitted from the response data set and the ALL option is included in the PROC GMAP statement.

The default outline color is the same as the default COUTLINE= color.

See also: ALL on page 740 and "Displaying Map Areas and Response Data" on page 738

COUTLINE=*nonempty-area-outline-color* | SAME
outlines non-empty map areas in the specified color. SAME specifies that the outline color of a map area is the same as the interior pattern color.

The default outline color depends on the PATTERN statement:

- □ If no PATTERN statement is specified, the default outline color is the foreground color (the first color in the colors list).
- □ If a PATTERN statement or the V6COMP graphics option is specified, the default is COUTLINE=SAME.

Note: If you specify empty map patterns, (VALUE=MEMPTY in a PATTERN statement) you should not change the outline color from the default value, SAME, to a single color. Otherwise all the outlines will be one color and you will not be able to distinguish between the empty areas. △

Featured in: Example 4 on page 778

CTEXT=*text-color*
specifies a color for the text in the legend. If you omit the CTEXT= option, a color specification is searched for in this order:

1 the CTEXT= option in a GOPTIONS statement
2 the default, the first color in the colors list.

The CTEXT= color specification is overridden if you also use the COLOR= suboption of a LABEL= or VALUE= option in a LEGEND definition that is assigned to the map legend. The COLOR= suboption determines the color of the legend label or the color of the legend value descriptions, respectively.

DESCRIPTION='*entry-description*'
DES='*entry-description*'
specifies a description of the catalog entry for the map. The maximum length for *entry-description* is 40 characters. The description does not appear on the map. By default, the GMAP procedure assigns a description of the form CHOROPLETH MAP OF *variable*, where *variable* is the name of the map variable.

Featured in: Example 5 on page 779

DISCRETE

treats a numeric response variable as a discrete variable rather than as a continuous variable. When you use the DISCRETE option, the response variables are not grouped into ranges; instead, the GMAP procedure uses a separate response level (pattern and color combination) for each different value of the formatted response variable. The LEVELS= option is ignored when you use the DISCRETE option.

Use this option if your numeric response variable is assigned a user-written format.

Note: If the data do not contain a value in a particular range of the format, that formatted range is not displayed in the legend. △

Featured in: Example 5 on page 779

HTML=*variable*

identifies the variable in the input data set whose values create links in the HTML file created by the ODS HTML statement. These links are associated with an area of the chart and point to the data or graph you wish to display when the user drills down on the area.

Featured in: Example 5 on page 779

HTML_LEGEND=*variable*

identifies the variable in the input data set whose values create links in the HTML file created by the ODS HTML statement. These links are associated with a legend value and point to the data or graph you wish to display when the user drills down on the value.

Featured in: Example 5 on page 779

LEGEND=LEGEND<1...99>

assigns the specified LEGEND definition to the map legend. LEGEND= is ignored if the specified LEGEND definition is not currently in effect. In the GMAP procedure, the CHORO statement produces a legend unless you use the NOLEGEND option. If you use the SHAPE= option in a LEGEND statement, only the value BAR is valid.

See also: "LEGEND Statement" on page 187

Featured in: Example 2 on page 773

LEVELS=*number-of-response-levels*

specifies the number of response levels to be graphed when the response variables are continuous. Each level is assigned a different combination of color and fill pattern.

If you do not use the LEVELS= option or the DISCRETE option, the GMAP procedure determines the number of response levels that use the formula FLOOR(1+3.3 log(*N*)), where *N* is the number of unique map area identification variable values.

The LEVELS= option is ignored when you use the DISCRETE or MIDPOINTS= option.

Featured in: Example 2 on page 773

MIDPOINTS=*value-list*

specifies the response levels for the range of response values that are represented by each level (pattern and color combination).

For numeric response variables, *value-list* is either an explicit list of values, or a starting and an ending value with an interval increment, or a combination of both forms:

n <...*n*>

n TO *n* <BY *increment* >

n <...*n*> TO *n* <BY *increment* > <*n* <...*n*> >

By default the increment value is 1. You can specify discrete numeric values in any order. In all forms, *n* can be separated by blanks or commas. For example,

```
midpoints=(2 4 6)
midpoints=(2,4,6)
midpoints=(2 to 10 by 2)
```

If a numeric variable has an associated format, the specified values must be the *unformatted* values.

For character response variables, *value-list* is a list of unique character values enclosed in quotation marks and separated by blanks:

'value-1' <...'value-n'>

The values are character strings enclosed in single quotation marks and separated by blanks. For example,

```
midpoints='Midwest' 'Northeast' 'Northwest'
```

Specify the values in any order. If a character variable has an associated format, the specified values must be the *formatted* values.

You can selectively exclude some response variable values from the map, as shown here:

```
midpoints='Midwest'
```

Only those observations for which the response variable exactly matches one of the values that are listed in the MIDPOINTS= option are shown on the map. As a result, observations may be excluded inadvertently if values in the list are misspelled or if the case does not match exactly.

Featured in: Example 8 on page 790

MISSING

accepts a missing value as a valid level for the response variable.

See also: "Displaying Map Areas and Response Data" on page 738

NAME='*entry-name*'

specifies the name of the catalog entry for the map. The maximum length for *entry-name* is eight characters. The default name is GMAP. If the specified name duplicates the name of an existing entry, SAS/GRAPH software adds a number to the duplicate name to create a unique entry, for example, GMAP1.

Featured in: Example 5 on page 779

NOLEGEND

suppresses the legend.

Featured in: Example 6 on page 786

XSIZE=*map-width* <*units*>
YSIZE=*map-height* <*units*>

specify the physical dimensions of the map that is to be drawn, where *n* is the number of units. By default, the map uses the entire procedure output area.

Valid *units* are CM (centimeters), IN (inches), or PCT (percentage of the graphics output area). By default, the unit is character cells (CELLS).

If you specify values for *n* that are greater than the dimensions of the procedure output area, the map is drawn using the default size.

If you specify either the XSIZE= or YSIZE= option without specifying the other option, the GMAP procedure rescales the dimension for the option that was not specified to retain the original shape of the map.

PRISM Statement

Creates three-dimensional prism maps in which levels of magnitude of the specified response variables are represented by polyhedrons (raised polygons) of varying height, pattern, and color.

Requirements: At least one response variable is required. You must use the ID statement in conjunction with the PRISM statement.

Global statements: FOOTNOTE, LEGEND, PATTERN, TITLE

Description

The PRISM statement specifies the variable or variables that contain the data that are represented on the map by raised map areas. This statement automatically

- determines the midpoints
- assigns patterns to the map areas.

You can use statement options to control the ranges of the response values, specify the angle of view, and enhance the appearance of the map.

In addition, you can use global statements to modify the map area patterns and the legend, as well as add titles and footnotes to the map. You can also use an Annotate data set to enhance the map.

For maps that contain intersecting polygons or polygons within polygons, extremely complicated maps, or maps that contain line segments that cross, use the GREDUCE procedure to reduce and simplify the map if necessary.

PRISM *response-variable(s)* <*/ option(s)*>;

option(s) can be one or more options from any or all of the following categories:

- appearance options:
  - ANNOTATE=*Annotate-data-set*
  - CEMPTY=*empty-area-outline-color*
  - COUTLINE=*nonempty-area-outline-color* | SAME
  - XLIGHT=*x*
  - YLIGHT=*y*
  - XSIZE=*map-width* <*units*>
  - YSIZE=*map-height* <*units*>
  - XVIEW=*x*
  - YVIEW=*y*
  - ZVIEW=*z*
- mapping options:
  - DISCRETE
  - LEVELS=*number-of-response-levels*
  - MIDPOINTS=*value-list*
  - MISSING
- legend options:
  - CTEXT=*text-color*
  - LEGEND=LEGEND<1...99>
  - NOLEGEND

- description options:
  - DESCRIPTION='*entry-description*'
  - NAME='*entry-name*'
- ODS options
  - HTML=*variable*
  - HTML_LEGEND=*variable*

Required Arguments

response-variable(s)

specifies one or more variables in the response data set that contains response values represented on the map. Each response variable produces a separate map. All variables must be in the input data set. Separate multiple response variables with blanks.

Missing values for the response variable are not considered valid unless you use the MISSING option in the PRISM statement.

Response variables can be either numeric or character. Numeric response variables with continuous values are grouped into ranges, or response levels. Each response level is assigned a different prism height and a different pattern and color combination. Character variables and numeric variables (when you use the DISCRETE option) have a different response level for each unique response variable value. This means that the prism height can be used to identify discrete values, but prism height does not reflect the specific value. Use the legend to determine the exact value of a discrete variable. Numeric variables are treated as continuous unless you use DISCRETE.

For numeric response variables with continuous values, you can control the selection of response level ranges using the MIDPOINTS= or LEVELS= option. By default, the GMAP procedure determines the number of levels for the map using the formula FLOOR(1+3.3 log(*N*)), where *N* is the number of unique map area identification variable values.

See also: "About Response Variables" on page 737

Options

Options in a PRISM statement affect all graphs that are produced by that statement. You can specify as many options as you want and list them in any order.

ANNOTATE=*Annotate-data-set*
ANNO=*Annotate-data-set*

specifies a data set to annotate maps that are produced by the PRISM statement.

Note: Annotate coordinate systems 1, 2, 7, and 8 are not valid with prism maps. △

See also: Chapter 10, "The Annotate Data Set," on page 403

CEMPTY=*empty-area-outline-color*

outlines empty map areas in the specified color. This option affects the map areas that are empty. Empty map areas are generated in prism maps either

- when there is no response value for a map area and the MISSING option is not used, or
- when a map area is omitted from the response data set and the ALL option is included in the PROC GMAP statement.

The default outline color is the same as the default COUTLINE= color.

See also: ALL on page 740 and “Displaying Map Areas and Response Data” on page 738

COUTLINE=*nonempty-area-outline-color* | SAME

outlines non-empty map areas in the specified color. SAME specifies that the outline color of a map area is the same as the interior pattern color.

The default outline color depends on the PATTERN statement:

- □ If no PATTERN statement is specified, the default outline color is the foreground color (the first color in the colors list).
- □ If a PATTERN statement or the V6COMP graphics option is specified, the default is COUTLINE=SAME.

Note: If you specify empty map patterns, (VALUE=MEMPTY in a PATTERN statement) you should not change the outline color from the default value, SAME, to a single color. Otherwise all the outlines will be one color and you will not be able to distinguish between the empty areas. △

Featured in: Example 7 on page 788

CTEXT=*text-color*

specifies a color for the text in the legend. If you omit the CTEXT= option, a color specification is searched for in this order:

1. the CTEXT= option in a GOPTIONS statement
2. the default, the first color in the colors list.

The CTEXT= color specification is overridden if you also use the COLOR= suboption of a LABEL= or VALUE= option in a LEGEND definition assigned to the map legend. The COLOR= suboption determines the color of the legend label or the color of the legend value descriptions, respectively.

DESCRIPTION=’*entry-description*’
DES=’*entry-description*’

specifies the description of the catalog entry for the chart. The maximum length for *entry-description* is 40 characters. The description does not appear on the chart. By default, the GMAP procedure assigns a description of the form PRISM MAP OF *variable*, where *variable* is the name of the map variable.

DISCRETE

treats a numeric response variable as a discrete variable rather than as a continuous variable. The DISCRETE option does not group the response values into ranges; instead, the GMAP procedure uses a separate response level (prism height, color, and surface pattern) for each different value of the formatted response variable. The LEVELS= option is ignored when you use the DISCRETE option.

Use this option if your numeric response variable is assigned a user-written format.

Note: If the data do not contain a value in a particular range of the format, that formatted range is not displayed in the legend. △

Featured in: Example 5 on page 779 (with CHORO statement)

HTML=*variable*

identifies the variable in the input data set whose values create links in the HTML file created by the ODS HTML statement. These links are associated with an area of the chart and point to the data or graph you wish to display when the user drills down on the area.

HTML_LEGEND=*variable*

identifies the variable in the input data set whose values create links in the HTML file created by the ODS HTML statement. These links are associated with a legend value and point to the data or graph you wish to display when the user drills down on the value.

LEGEND=LEGEND<1...99>

assigns the specified LEGEND definition to the map legend. LEGEND= is ignored if the specified LEGEND definition is not currently in effect. In the GMAP procedure, the PRISM statement produces a legend unless you use the NOLEGEND option. If you use the SHAPE= option in a LEGEND statement, only the value BAR is valid.

See also: "LEGEND Statement" on page 187

Featured in: Example 8 on page 790

LEVELS=*number-of-response-levels*

specifies the number of response levels to be graphed when the response variables are continuous. Each level is assigned a different prism height, surface pattern, and color combination.

If neither the LEVELS= option nor the DISCRETE option is used, the GMAP procedure determines the number of response levels that use the formula FLOOR(1+3.3 log(*N*)), where *N* is the number of unique map area identification variable values.

The LEVELS= option is ignored when you use the DISCRETE or MIDPOINTS= option.

Featured in: Example 2 on page 773

MIDPOINTS=*value-list*

specifies the response levels for the range of response values that are represented by each level (prism height, pattern, and color combination).

For numeric response variables, *value-list* is either an explicit list of values, or a starting and an ending value with an interval increment, or an combination of both forms:

n <...*n*>

n TO *n* <BY *increment*>

n <...*n*> TO *n* <BY *increment* > <*n* <...*n*> >

By default the increment value is 1. You can specify discrete numeric values in any order. In all forms, *n* can be separated by blanks or commas. For example,

```
midpoints=(2 4 6)
midpoints=(2,4,6)
midpoints=(2 to 10 by 2)
```

If a numeric variable has an associated format, the specified values must be the *unformatted* values.

For character response variables, *value-list* has this form:

'value-1' <...'value-n'>

The values are character strings enclosed in single quotation marks and separated by blanks. For example,

```
midpoints='Midwest' 'Northeast' 'Northwest'
```

Specify the values in any order. If a character variable has an associated format, the specified values must be the *formatted* values.

You can selectively exclude some response variable values from the map, as shown here:

```
midpoints='Midwest'
```

Only those observations for which the response variable exactly matches one of the values listed in the MIDPOINTS= option are shown on the map. As a result, observations may be inadvertently excluded if values in the list are misspelled or if the case does not match exactly.

Featured in: Example 8 on page 790

MISSING

accepts a missing value as a valid level for the response variable.

See also: "Displaying Map Areas and Response Data" on page 738

NAME='*entry-name*'

specifies the name of the catalog entry for the map. The maximum length for *entry-name* is eight characters. The default name is GMAP. If that name that you specify duplicates the name of an existing entry, SAS/GRAPH software adds a number to the duplicate name to create a unique entry, for example, GMAP1.

NOLEGEND

suppresses the legend.

XLIGHT=*x*
YLIGHT=*y*

specify the coordinates of the imagined light source in the map coordinate system. The position of the light source affects the way the sides of the map polygons are shaded. Although you can specify any point for the light source using the XLIGHT= and YLIGHT= options, the light source is actually placed in one of only four positions.

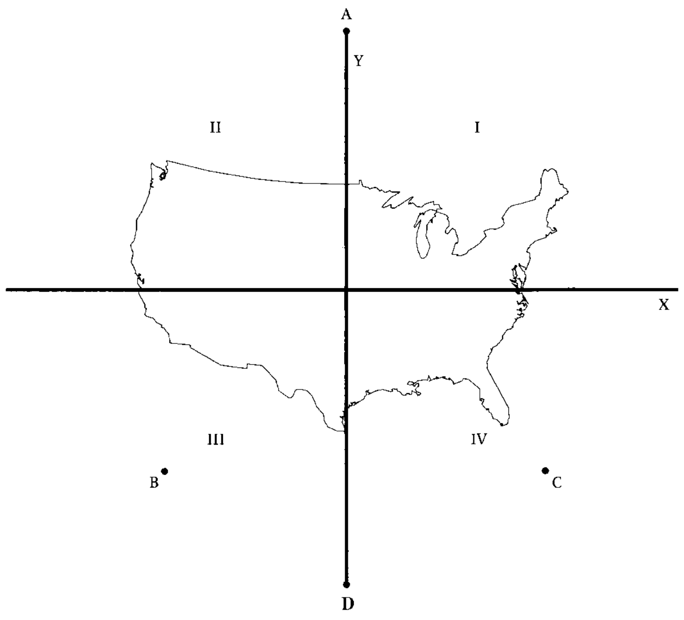
Table 19.1 on page 757 shows how the point you specify is positioned.

Table 19.1 Light Source Coordinates

Specified Light Source	Light Source Position
in quadrants I or II, or on the X or +Y axis	behind the map (point A), and all side polygons are shadowed
on or within approximately 10 degrees of the Y axis	the viewing position (point D), and none of the side polygons are shadowed
in quadrant III (except within 10 degrees of the Y axis)	to the left of the map (point B), and the right-facing sides of polygons are shadowed
in quadrant IV (except within 10 degrees of the Y axis)	to the right of the map (point C), and the left-facing side polygons are shadowed

Figure 19.6 on page 758 illustrates the light source positions. Assume that your viewing position, selected by the XVIEW=, YVIEW=, and ZVIEW= options, is point D.

Figure 19.6 Coordinates of Imagined Light Source in a Map Coordinate System



By default, the light source position is the same as the viewing position specified by the XVIEW=, YVIEW=, and ZVIEW= options. The light source position cannot coincide with the viewing reference point (0.5,0.5), which corresponds with the position directly above the center of the map.

See also: XVIEW= on page 758

Featured in: Example 8 on page 790

XSIZE=*map-width* <*units*>
YSIZE=*map-height* <*units*>

specify the physical dimensions of the map that is to be drawn, where *n* is the number of units. By default, the map uses the entire procedure output area.

Valid *units* are CM (centimeters), IN (inches), or PCT (percentage of the graphics output area). By default, the unit is character cells (CELLS).

If you specify values for *map-width* and *map height* that are greater than the dimensions of the procedure output area, the map is drawn using the default size. And if you specify one value and not the other, the dimension is adjusted to maintain the correct aspect ratio.

XVIEW=*x*
YVIEW=*y*
ZVIEW=*z*

specify the viewing position coordinates for the map. In this system, the four corners of the map lie on the X-Y plane at coordinates (0,0,0), (0,1,0), (1,1,0), and (1,0,0). No axes are actually drawn on the maps that are produced by PROC GMAP, but imagine that the maps are drawn in an X-Y plane.

Your viewing position cannot coincide with the viewing reference point at coordinates (0.5,0.5,0), the center of the map. The value for *z* cannot be negative.

If you omit the XVIEW=, YVIEW=, and ZVIEW= options, the default coordinates are (0.5,–2,3). This viewing position is well above and to the south of the center of the map. Specify one, two, or all three of the view coordinates; any that you do not explicitly specify are assigned the default values.

Figure 19.5 on page 747 shows the position of the viewing reference point, as well as the default viewing position.

To ensure that the polygon edges are distinguishable, the angle from vertical must be less than or equal to 45 degrees. If you specify a ZVIEW= value such that this condition cannot be satisfied (that is, a very small value), PROC GMAP increases the ZVIEW= value automatically so that the angle is 45 degrees or less.

Featured in: Example 8 on page 790

SURFACE Statement

Creates three-dimensional surface maps in which levels of magnitude of the specified response variables are represented by spikes of varying height.

Requirements: At least one response variable is required and must be numeric. The ID statement must be used in conjunction with the SURFACE statement.

Global statements: FOOTNOTE, TITLE

Description The SURFACE statement specifies the variable or variables that contain the data that are represented on the map by raised map areas. This statement automatically determines the midpoints. You can use statement options to control spike proportions, specify the angle of view, and modify the general appearance of the map. For example, you can select the color and number of lines for the representation of the surface area. You can control the selection of spike heights and base widths.

In addition, you can use global statements to add titles and footnotes to the map. You can also use an Annotate data set to enhance the map.

SURFACE*response-variable(s)* </ *option(s)*>;

option(s) can be one or more options from any or all of the following categories:

- appearance options:
  - ANNOTATE=*Annotate-data-set*
  - CBODY=*surface-map-color*
  - CONSTANT=*n*
  - NLINES=*number-of-lines*
  - ROTATE=*degrees*
  - TILT=*degrees*
  - XSIZE=*map-width <units>*
  - YSIZE=*map-height <units>*
- description options:
  - DESCRIPTION=’*entry-description*’
  - NAME=’*entry-name*’

Required Arguments

response-variable(s)
specifies one or more variables in the response data set that contains response values for map areas in the map data set. *Response-variable* must be numeric and must contain only positive values. Each response variable produces a separate map. All variables must be in the input data set. Separate multiple response variables with blanks.

The GMAP procedure scales response variables for presentation on the map. The height of the spikes on the map correspond to the relative value of the response variable, not to the actual value of the response variable. However, when the viewing angle is changed, the spikes may not appear this way. The spikes in the front may appear to be higher than the spikes in the back, which represent greater values.

See also: "About Response Variables" on page 737

Options

SURFACE statement options affect all maps that are produced by that statement.

ANNOTATE=*Annotate-data-set*
ANNO=*Annotate-data-set*
specifies a data set to annotate maps that are produced by the SURFACE statement.

Note: Annotate coordinate systems 1, 2, 7, and 8 are not valid with surface maps. △

See also: Chapter 10, "The Annotate Data Set," on page 403

CBODY=*surface-map-color*
specifies the color that is used to draw the surface map. By default, the first color in the current colors list is used.

CONSTANT=*n*
specifies a denominator to use in the distance decay function. This function determines the base width of the spike that is drawn at each map area center.

By default, CONSTANT=10. Values greater than 10 yield spikes that are wider at the base. Values less than 10 yield spikes that are narrower at the base.

Let x_k and y_k represent the coordinates, and z_k represent the function value at the center of each map area. The z_k values are scaled from 1 to 11. A square grid of *x* by *y* points (where the size of the grid is the NLINES= option value) and the associated function value *f(x,y)* are generated from the map area center value using this formula:

$$f\left(x,y\right)=\sum^{k}\left(1-1.5^{k}+.5\mathrm{D}^{3k}\right)\triangle^{k\mathrm{zk}}$$

where

$$\mathrm{D}^{k}=\left(x-x^{k}\right)^{2}+\left(y-y^{k}\right)^{k}$$

and

$$\triangle^{k}=\left[\mathrm{martix\,cdelim}=\mathrm{XXXXXXXXXXXXXXX}\,1\,\mathrm{if}\,\mathrm{D}^{k}<1,\,0\,\mathrm{otherwise.}\right]$$

Featured in: Example 10 on page 792

DESCRIPTION='*entry-description*'
DES='*entry-description*'
specifies the description of the catalog entry for the map. The maximum length for *entry-description* is 40 characters. The description does not appear on the map. By default, the GMAP procedure assigns a description of the form SURFACE MAP OF *variable*, where *variable* is the name of the map variable.

NAME='*entry-name*'
specifies the name of the catalog entry for the map. The maximum length for *entry-name* is eight characters. The default name is GMAP. If the specified name duplicates the name of an existing entry, SAS/GRAPH software adds a number to the duplicate name to create a unique entry, for example, GMAP1.

NLINES=*number-of-lines*
N=*number-of-lines*
specifies the number of lines, *n*, used to draw the surface map. Values for *n* are 50 to 100; the higher the value, the more solid the map appears and the more resources used. By default, NLINES=50.

Featured in: Example 10 on page 792

ROTATE=*degrees*
specifies the degrees of the angle at which to rotate the map about the Z axis in the map coordinate system. *Degrees* can be any angle. Positive values indicate rotation in the counterclockwise direction. By default, ROTATE=70. The ROTATE= option also affects the direction of the lines that are used to draw the surface map.

Featured in: Example 10 on page 792

TILT=*degrees*
specifies the degrees of the angle at which to tilt the map about the X axis in the map coordinate system. *Degrees* can be 0 to 90. Increasing values cause the map to tilt backward and makes the spikes more prominent. Decreasing values make the map shape more distinguishable and the spikes less prominent. TILT=90 corresponds to viewing the map edge-on, while TILT=0 corresponds to viewing the map from directly overhead. By default, TILT=70.

Featured in: Example 10 on page 792

XSIZE=*map-width* <*units*>
YSIZE=*map-height* <*units*>
specify the physical dimensions of the map to be drawn, where *n* is the number of units. By default, the map uses the entire procedure output area.

Valid *units* are CM (centimeters), IN (inches), or PCT (percentage of the graphics output area). By default, the unit is character cells (CELLS).

If you specify values for *map-width* and *map-height* that are greater than the dimensions of the procedure output area, the map is drawn using the default size. And if you specify only one dimension, the other is scaled to maintain the aspect ratio.

SAS/GRAPH Map Data Sets

Many map data sets are provided with SAS/GRAPH software. Some of the more commonly used map data sets are listed in the following table (see the MAPS.METAMAPS data set for details on all the Institute-supplied map data sets). The items listed under each of the variable names describes the value type. For example, *proj* means that the value for the specified variable is a projected value and *long* or *lat* means that the value is unprojected longitude or latitude. If a variable does not exist in the map data set, *n/a* is used in the table.

Note: The last two items in the table (Continents and International) are categories of map data sets. The map data sets in these categories contain individual countries, continents, or subdivisions. To see the complete list of map data sets that are provided in each of these categories, see the MAPS.METAMAPS data set. △

Table 19.2 SAS/GRAPH Map Data Sets

Version 6 Variables			Current Variables				Compatible with V6?		
Map Data Sets	X	Y	LONG	LAT	X	Y	LONG	LAT	
US	proj	proj	n/a	n/a	proj	proj	n/a	n/a	yes
USCITY[1]	proj	proj	long	lat	proj	proj	long	lat	yes[2]
USCENTER	proj	proj	long	lat	proj	proj	long	lat	yes
STATES	long	lat	n/a	n/a	long	lat	n/a	n/a	yes
COUNTIES	long	lat	n/a	n/a	long	lat	n/a	n/a	yes
COUNTY	long	lat	n/a	n/a	long	lat	n/a	n/a	yes
USCOUNTY	proj	proj	n/a	n/a	proj	proj	n/a	n/a	yes
CANADA	proj	proj	n/a	n/a	proj	proj	n/a	n/a	yes
CANADA2	proj	proj	n/a	n/a	proj	proj	n/a	n/a	yes
CANADA3	long	lat	n/a	n/a	long	lat	n/a	n/a	yes
CANADA4	long		n/a	n/a	long	lat	n/a	n/a	yes
WORLDPRJ	proj	proj	n/a	n/a	not available in current version	not available in current version	not available in current version	not available in current version	no
WORLDMAP	long	lat	n/a	n/a	not available in current version	not available in current version	not available in current version	not available in current version	no
WORLD	not available in Version 6	not available in Version 6	not available in Version 6	not available in Version 6	proj	proj	long	lat	no
Continents	proj	proj	n/a	n/a	proj	proj	long	lat	yes
International	proj	proj	n/a	n/a	proj	proj	long	lat	yes[3]

1 The USCITY and USCENTER map data sets contain coordinates for labeling (see “Specialty Map Data Sets” on page 736 for details)
2 Contact Technical Support for a program that can be used to create the data sets from MAPS.WORLD.
3 The Version 6 data sets that contained multiple countries have been divided into individual data sets, beginning in Version 7.

Locating Map Data Sets

A collection of map data sets is supplied with SAS/GRAPH. Contact your SAS Support Consultant to verify the name and location of the SAS data library that contains the map data sets at your site before you use the map data sets. Many sites automatically assign a libref of MAPS to the SAS data library that contains the Institute-supplied map data sets. However, if you use the map data sets regularly and your site does not automatically assign a libref to the data library that contains the map data sets, you can add a LIBNAME statement to your AUTOEXEC file that defines the location of the map data set library. If you do this, the libref for the maps is established automatically whenever you begin a SAS session.

Accessing Descriptions of Map Data Sets

You may need detailed information on the map data sets in order to determine their size, the variables they contain, or whether they are projected or unprojected. You can get this information by using the CONTENTS or DATASETS procedure, or browsing the METAMAPS data set in the MAPS library (or the library where your Institute-supplied map data sets reside). For example, these statements list the map data sets in the SAS data library that is assigned to the libref MAPS:

```
libname maps 'SAS-data-library';

proc datasets lib=maps;
run;
```

Note: Be sure to replace *SAS-data-library* with the location of the SAS data library that contains map data sets at your site. △

The following statements provide detailed information on the map data sets, including the number of observations, the variables in each data set, and a description of each variable:

```
libname maps 'SAS-data-library';

proc contents data=maps.canada3;
run;
```

To see the contents and descriptions of all of the Institute-supplied map data sets you can specify DATA=MAPS._ALL_ in the CONTENTS procedure. See the *SAS Procedures Guide* for more information on the CONTENTS and DATASETS procedures.

Using FIPS Codes and Province Codes

The map area identification variable in some map data sets that are included with SAS/GRAPH contain standardized numeric codes. The data sets for the United States contain a variable whose values are FIPS (Federal Information Processing System) codes. The data sets for Canada contain standard province codes or census division codes. When you use the GMAP procedure, the variables that identify map areas in your response data set must have the same values as the map area identification variables in the map data set that you are using. If the map area identification variables in your response data set are state or province names or abbreviations, convert them to FIPS codes or province codes before using the response data set with one of the Institute-supplied map data sets. Table 19.3 on page 764 lists the FIPS codes for the United States and Table 19.4 on page 764 lists the standard codes for Canadian provinces.

Table 19.3 U.S. FIPS Codes

FIPS Code	State	FIPS Code	State
01	Alabama	30	Montana
02	Alaska	31	Nebraska
04	Arizona	32	Nevada
05	Arkansas	33	New Hampshire
06	California	34	New Jersey
08	Colorado	35	New Mexico
09	Connecticut	36	New York
10	Delaware	37	North Carolina
11	District of Columbia	38	North Dakota
12	Florida	39	Ohio
13	Georgia	40	Oklahoma
15	Hawaii	41	Oregon
16	Idaho	42	Pennsylvania
17	Illinois	44	Rhode Island
18	Indiana	45	South Carolina
19	Iowa	46	South Dakota
20	Kansas	47	Tennessee
21	Kentucky	48	Texas
22	Louisiana	49	Utah
23	Maine	50	Vermont
24	Maryland	51	Virginia
25	Massachusetts	53	Washington
26	Michigan	54	West Virginia
27	Minnesota	55	Wisconsin
28	Mississippi	56	Wyoming
29	Missouri	72	Puerto Rico

Table 19.4 Canadian Province Codes

Province Code	Province
10	Newfoundland
11	Prince Edward Island
12	Nova Scotia
13	New Brunswick

Province Code	Province
24	Quebec
35	Ontario
46	Manitoba
47	Saskatchewan
48	Alberta
59	British Columbia
60	Yukon
61	Northwest Territories

Note: The ID variables in Canadian maps are character. △

The CNTYNAME data set contains a cross-reference of names and FIPS codes for all counties in the United States. The CANCENS data set contains a cross-reference of census district names and codes for Canadian provinces.

Base SAS software provides several functions that convert state names to FIPS codes and vice versa. The following table lists these functions and a brief description of each. See *SAS Language Reference: Dictionary* for more information.

Table 19.5 FIPS and Postal Code Functions

Function	Description
STFIPS	converts state postal code to FIPS state code
STNAME	converts state postal code to state name in upper case
STNAMEL	converts state postal code to state name in mixed case
FIPNAME	converts FIPS code to state name in upper case
FIPNAMEL	converts FIPS code to state name in mixed case
FIPSTATE	converts FIPS code to state postal code

Using SAS/GRAPH Map Data Sets

You can customize the area that is displayed on your map by using only part of a particular map data set. There are several ways to accomplish this. You can use WHERE processing or a DATA step to subset the map data to be used by the GMAP procedure. You can also use the GPROJECT procedure to create a rectangular subset of a map data set by using minimum and maximum longitude and latitude values.

You can combine map data sets in either of these situations:

- The map data sets to be combined were originally projected together.
- The map data sets all contain the same type of coordinates. That is, all are in radians or all are in degrees.

Institute-supplied map data sets that have coordinates expressed only as longitude and latitude, with variable names LONG and LAT, must be renamed X and Y and should be projected before displaying.

Subsetting Map Data Sets

Some of the map data sets that are included with SAS/GRAPH contain a large number of observations. Programs that use only a few states or provinces will run faster if you exclude the unused portion of the map data set or use an already reduced map data set. The SAS System provides several ways to accomplish this. One is to use the WHERE statement or WHERE= data set option within the GMAP procedure to select only the states or provinces you want.

For example, to use only the observations for Quebec in the CANADA data set, begin the GMAP procedure with this statement:

```
proc gmap map=maps.canada(where=(province='24'));
```

If you use the WHERE statement, the WHERE condition applies to both the map data set and the response data sets. The WHERE= data set option applies only to the data set that you specify in the argument in which the WHERE= option appears.

The WHERE statement and WHERE= data set option are most useful when you produce a simple map and do not need to make any other changes to the data set.

Another approach is to use a DATA step to create a subset of the larger data set. This code illustrates another way to extract the observations for Quebec from the CANADA data set:

```
data quebec;
   set maps.canada(where=(province='24'));
```

This approach is most useful when you want to create a permanent subset of a map data set or when you need to make additional changes to the map data set.

Also see Chapter 25, "The GREMOVE Procedure," on page 905 for an example how to use GREMOVE to create a regional map from one of the data sets that are supplied with SAS/GRAPH.

Reducing Map Data Sets

You can reduce map data sets. A *reduced map data set* is one that can be used to draw a map that retains the overall appearance of the original map but that contains fewer points, requires considerably less storage space, and can be drawn much more quickly. You can improve performance by plotting fewer observations for each map area. You reduce a map data set when you subset it on the variable DENSITY. You can add the variable DENSITY to a map data set by using the GREDUCE procedure. For more information, see Chapter 24, "The GREDUCE Procedure," on page 895.

An *unreduced map data set* contains all of the coordinates that were produced when the map was digitized. This type of map data set has more observations than most graphics output devices can accurately plot. Some unreduced map data sets already contain a DENSITY variable like the one calculated by the GREDUCE procedure, so it is not necessary to use the GREDUCE procedure to process these data sets. Values for DENSITY range from 0 through 6 (the lower the density, the coarser the boundary line).

A statement of this form excludes all points with a density level of 2 or greater:

```
proc gmap map=maps.states(where=(density<2));
```

The resulting map is much coarser than one drawn by using all of the observations in the data set, but it is drawn much faster.

Another way to create a reduced map data set is to use a DATA step to exclude observations with larger density values:

```
data states;
   set maps.states(where=(density<2));
```

Projecting Map Data Sets

Map data can be stored as unprojected or projected coordinates. Unprojected map data contains spherical coordinates, that is, longitude and latitude values usually expressed in radians. * A few map data sets that are provided with SAS/GRAPH contain only unprojected coordinates and should be projected before you use them. They are

CANADA3

CANADA4

COUNTIES

COUNTY

STATES

Projected map data contains Cartesian coordinates. The GMAP procedure is designed to plot maps by using projected map data sets. Most SAS/GRAPH map data sets contain projected coordinates that are stored as X and Y. If the projection supplied with the map data set does not meet your needs, you can use the GPROJECT procedure to create a different projection. You should select a projection method that least distorts the regions that you are mapping. (All projection methods inherently distort map regions.) See Chapter 23, "The GPROJECT Procedure," on page 873 for more information.

Note: Using an unprojected map data set with the GMAP procedure can cause your map to be reversed and distorted. △

Controlling the Display of Lakes

Some countries contain a lake that is located completely within a single unit area. Occasionally these lakes can be a problem. For example, displaying lakes in prism maps may cause undesirable results. In addition, displaying lakes may not be appropriate for some applications. In these cases, you may want to remove the lakes from the map data set before you proceed.

Map data sets that contain coordinates for a lake that is located within a single internal division are identified by the presence of the character variable LAKE. The value of LAKE is 1 for points that correspond to lakes and 0 otherwise. The following statements illustrate how to delete the lakes from your map data sets using WHERE processing:

```
proc gmap map=maps.chile(where=(lake='0'))
          data=maps.chile;
   id id;
   choro id / levels=1 nolegend;
   title box=1 f=none h=4
         'Chile with Lakes Removed';
run;
```

You can also create a new map data set that is a subset of the map data set:

```
data nolake;
   set maps.chile;
   if lake='0';
run;
```

* If your data is in degrees, it can be converted to radians by multiplying by the degree-to-radian constant [atan(1)/45].

Creating Map Data Sets

In addition to using map data sets that are supplied with SAS/GRAPH software, you may want to create your own map data sets. Map data sets are not limited to geographic data; you use them to define other spaces such as floor plans or street diagrams. This section explains more about the structure of map data sets.

A unit area is defined by observations in the map data set that have the same identification (ID) variable value. A unit area may be composed of a single polygon or a collection of polygons. A polygon is defined by all of the observations that have the same SEGMENT variable value.

- If the unit area is a single polygon, all values of SEGMENT are the same.
- If the unit area contains multiple polygons, such as islands, the SEGMENT variable has multiple values. For example, in the MAPS.US data set, the state of Hawaii (a unit area) contains six different values in the SEGMENT variable, one for each island in the state.
- If the unit area contains enclosed polygons, such as lakes, the SEGMENT variable has one value but the interior polygon is defined by separate boundaries. For example, in the CANADA2 data set supplied with SAS/GRAPH, the map data for the Northwest Territories (a unit area) use enclosed polygons for two lakes.

Creating a unit area that is a single polygon.

This DATA step creates a SAS data set that contains coordinates for a unit area with a single polygon, a square:

```
data square;
   input id x y;
   datalines;
1 0 0
1 0 40
1 40 40
1 40 0
;
```

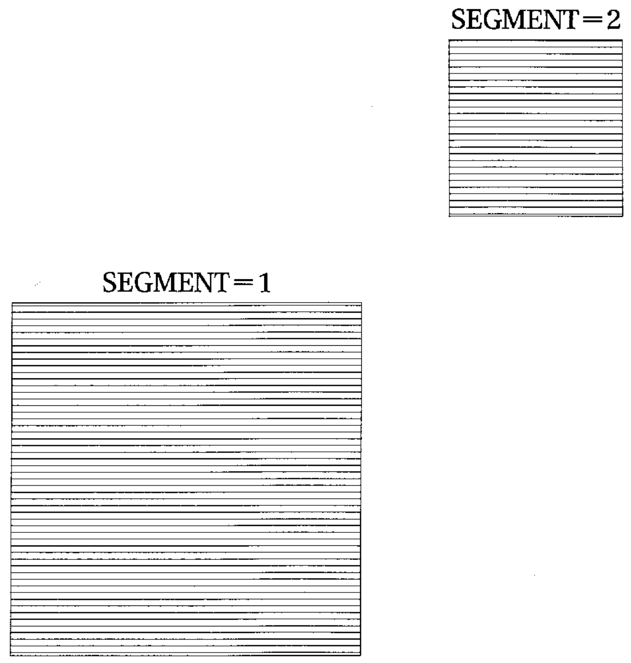
This data set does not have a SEGMENT variable.

Creating a unit area that contains multiple polygons.

Use different values of the SEGMENT variable to create separate polygons within a single unit area. For example, this DATA step assigns two values to the SEGMENT variable. The resulting data set produces a single unit area that contains two polygons, as shown in Figure 19.7 on page 769:

```
data map;
   input id $ 1-8 segment x y;
   datalines;
square   1 0 0
square   1 0 4
square   1 4 4
square   1 4 0
square   2 5 5
square   2 5 7
square   2 7 7
square   2 7 5
;
```

Figure 19.7 Single Unit Area with Two Segments (Polygons)

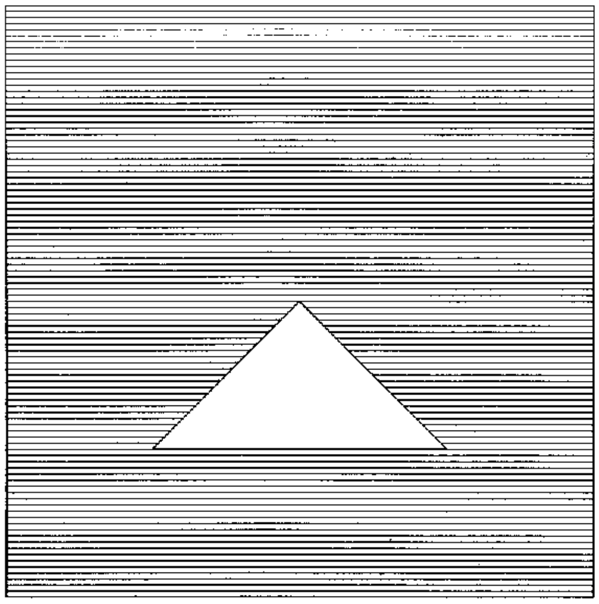


Creating a unit area that contains enclosed polygons as holes.

Use separate boundaries to create an enclosed polygon (that is, a polygon that falls within the primary polygon for a single segment). The separate boundaries are separated from the primary polygon boundary by missing values for X and Y. For example, the data set that is created by this DATA step produces the map shown in Figure 19.8 on page 770:

```
data map;
   input id $ 1-8 segment x y;
   datalines;
square   1 0 0
square   1 0 4
square   1 4 4
square   1 4 0
square   1 . .
square   1 1 1
square   1 2 2
square   1 3 1
;
```

Figure 19.8 Single Unit Area with Hole

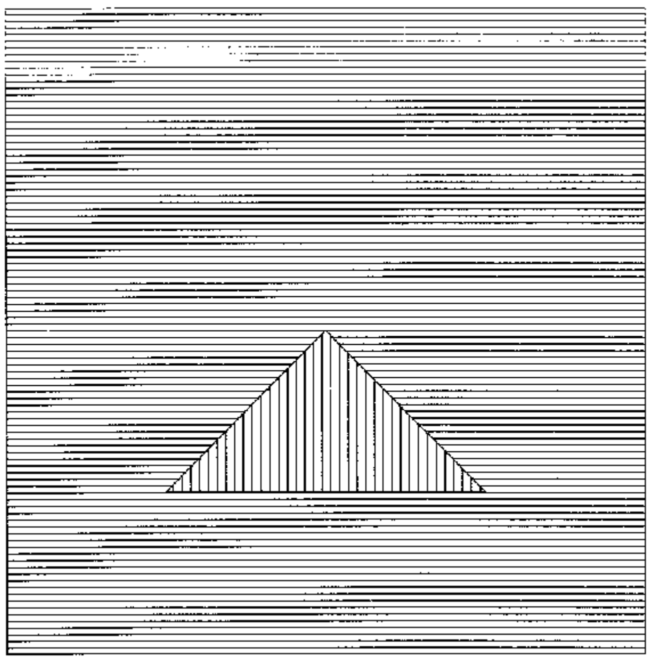


Creating a unit area that contains enclosed polygons as cities.

Ordinarily, if one unit area is surrounded by another, the pattern of the external unit area is drawn over the pattern for the internal one, instead of around it. Avoid this problem by adding an observation to the map data for the external unit area with missing values for X and Y, followed by the coordinates of the internal unit area, but using the ID values for the external unit area. For example, this DATA step creates a data set that produces the map shown in Figure 19.9 on page 771:

```
data map;
   input id $ 1-8 segment x y;
   datalines;
square   1 0 0
square   1 0 4
square   1 4 4
square   1 4 0
square   1 . .
square   1 1 1
square   1 2 2
square   1 3 1
triangle 1 1 1
triangle 1 2 2
triangle 1 3 1
;
```

Figure 19.9 Unit Area within a Unit Area



Note: A single map segment (a section of a unit area with a single value of the SEGMENT variable) cannot contain multiple polygons without at least one observation with missing values for X and Y. All segments within the map data sets that are supplied by SAS/GRAPH contain a single polygon that can have one or more separate boundaries, each separated by an observation with missing values for X and Y. △

Examples

The following examples include features from one or more of the GMAP statements.

Example 1: Producing a Simple Block Map

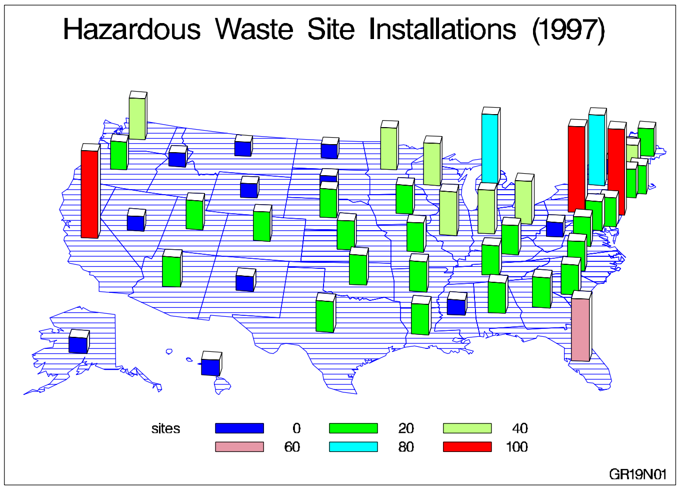
Procedure features:

ID statement

BLOCK statement option:

CBLKOUT=

Sample library member: GR19N01



This example produces a block map that shows the total number of hazardous waste sites in each state in 1997. Since the DISCRETE option is not used, the response variable is assumed to have a continuous range of values. Because neither the LEVELS= nor MIDPOINTS= option is used, the GMAP procedure selects a number of levels based on the number of map areas and then calculates the appropriate response levels. The legend shows the midpoint value of each level.

The blocks use the default pattern, which is a solid fill that rotates through the colors list. Because the colors list is specified in the GOPTIONS statement, all colors are used in the rotation. CBLKOUT= outlines the blocks in black, instead of using the default outline color, which is the first color in the list– in this case, BLUE.

The map areas use the default pattern for map areas in a block map. This is the first hatch pattern for maps, M2N0. By default, both the fill and the outline use the first color in the colors list.

Assign the librefs and set the graphics environment. COLORS= specifies the colors list, which is used by the default patterns and outlines. CTEXT= specifies the color for all text on the output.

```
libname reflib 'SAS-data-library';
libname maps 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
         colors=(blue green lime lipk cyan red)
         ctext=black ftext=swiss htitle=6 htext=3;
```

Create response data set REFLIB.SITES. This data set contains a map area identification variable, STATE, and a response variable, SITES. STATE contains the FIPS codes for each state and matches the values of STATE in the MAPS.US data set. SITES contains the total number of waste sites installed in the state.

```
data reflib.sites;
   length stcode $ 2;
   input region stcode $ sites;
   state=stfips(stcode);
   datalines;
6   AR  12
10  AK  7...moredata lines...
3   WV  6
```

```
8   WY  3
;
```

Define title and footnote for map.

```
title1 'Hazardous Waste Site Installations (1997)';
footnote1 j=r 'GR19N01 ';
```

Produce the block map. The ID statement specifies the variable that is in both the map data set and the response data set and defines map areas. The BLOCK statement specifies the variable in the response data set that contains the response values for each of the map areas. CBLKOUT= specifies the color for the block outlines.

```
proc gmap map=maps.us data=reflib.sites;
   id state;
   block sites / cblkout=black;
run;
quit;
```

Example 2: Specifying Response Levels in a Block Map

Procedure features:

BLOCK statement options:

- LEGEND=
- LEVELS=
- SHAPE=
- XVIEW=
- ZVIEW=

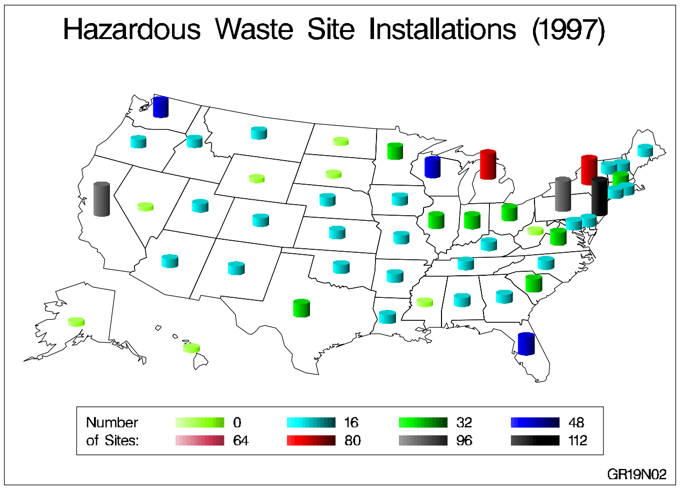
Other features:

LEGEND statement

PATTERN statement

Data set: REFLIB.SITES

Sample library member: GR19N02



This example uses LEVELS= to specify the number of response levels for the blocks. LEVELS= tells GMAP how many response levels and GMAP calculates the midpoints. Eight PATTERN statements explicitly define a color for each of these response levels.

A single PATTERN statement uses the REPEAT= option to define an empty map/plot pattern outlined in black for all the map areas.

The example also changes the viewpoint by rotating the map to provide a better view of the northeast states. As a result, the blocks appear shorter.

Assign the librefs and set the graphics environment.

```
libname reflib 'SAS-data-library';
libname maps 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
         colors=(black blue green red)
         ctext=black ftext=swiss htitle=6 htext=3;
```

Define title and footnote for map.

```
title1 'Hazardous Waste Site Installations (1997)';
footnote1 j=r 'GR19N02 ';
```

Define the patterns for the blocks. PATTERN statements 1-8 specify bar/block patterns and cannot be used by the map areas. They are applied to the blocks in order of the response level.

```
pattern1 value=solid color=lime;
pattern2 value=solid color=cyan;
pattern3 value=solid color=green;
pattern4 value=solid color=blue;
pattern5 value=solid color=lipk;
pattern6 value=solid color=red;
pattern7 value=solid color=gray;
pattern8 value=solid color=black;
```

Define a pattern for the map areas. PATTERN9 defines a single map pattern that is repeated for each of the 50 map areas (states). The pattern is an empty fill with a black border. VALUE= defines a map/plot pattern, which cannot be used by the blocks. Specifying a color causes PATTERN9 to generate only one pattern definition. REPEAT= specifies the number of times to repeat the pattern definition.

```
pattern9 value=mempty color=black repeat=50;
```

Define legend characteristics. LABEL= produces a two line label and places it to the left of the legend values. FRAME draws a border around the legend using the first color in the colors list.

```
legend1 value=(justify=left)
        label=('Number' justify=left 'of Sites:'
               position=(middle left))
        frame;
```

Produce the block map. LEVELS= specifies the number of response levels for the graph. SHAPE= draws the blocks as 3D cylinders. XVIEW= changes the viewpoint for the map so that the map appears to be slightly rotated. ZVIEW= raises the height of the viewpoint. LEGEND= assigns the LEGEND1 statement to the map legend.

```
proc gmap map=maps.us data=reflib.sites;
   id state;
   block sites / levels=8
                 shape=cylinder
                 xview=0.75
                 zview=5
                 legend=legend1;
run;
quit;
```

Example 3: Assigning a Format to the Response Variable

Procedure features:

BLOCK statement options:

- AREA=
- CBLKOUT=
- COUTLINE=
- DISCRETE

Other features:

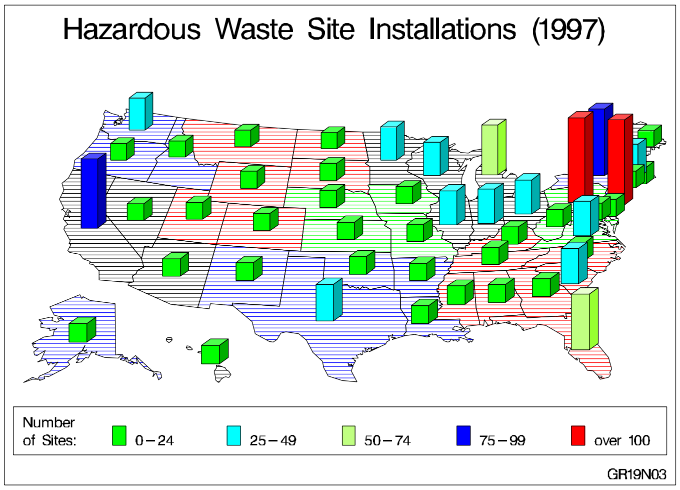
FORMAT statement

LEGEND statement

PATTERN statement

Data set: REFLIB.SITES

Sample library member: GR19N03



This example creates a format that defines the ranges of values for the response values and assigns this format to the response variable. These ranges appear in the legend and make the map easier to understand. When a format is assigned to a numeric response variable, the DISCRETE option must be used so that each formatted value is treated as a separate response level.

The example also patterns the map areas by region. To do this, both data sets must contain the ID variable, REGION. The response data set, REFLIB.SITES, already contains REGION, so the program only needs to add it to the map data set. Then the map data set is sorted by both the ID variables, REGION and STATE. Finally, the AREA= option specifies that the ID variable REGION is the one by which the map areas are patterned.

Assign the librefs and set the graphics environment.

```
libname reflib 'SAS-data-library';
libname maps 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
         colors=(black blue green red)
         ftext=swiss htitle=6 htext=3;
```

Create map data set REFLIB.STATES1 by adding REGION to the MAPS.US map data set.

```
data reflib.states1;
   set maps.us;
   select;
      when (state in (9,23,25,33,44,50))       region=1;
      when (state in (34,36))                  region=2;
      when (state in (10,11,24,42,51,54))      region=3;
      when (state in (1,12,13,21,28,37,45,47)) region=4;
      when (state in (17,18,26,27,39,55))      region=5;
      when (state in (5,22,35,40,48))          region=6;
      when (state in (19,20,29,31))            region=7;
      when (state in (8,30,38,46,49,56))       region=8;
      when (state in (4,6,15,32))              region=9;
      otherwise                                region=10;
   end;
```

```
run;
```

Sort the new map data set. The map data must be sorted in the order of the ID variables.

```
proc sort data=reflib.states1 out=reflib.states2;
   by region state;
run;
```

Create a format for SITES. SITESFMT. defines and labels the ranges of values for SITES.

```
proc format;
   value sitesfmt low-24='0-24'
                  25-49='25-49'
                  50-74='50-74'
                  75-99='75-99'
                  100-high='over 100';
run;
```

Define title and footnote for map.

```
title1 'Hazardous Waste Site Installations (1997)';
footnote j=r 'GR19N03 ';
```

Define a hatch pattern for the map areas. PATTERN1 defines a dense hatch pattern for the map areas. Because there are four colors in the colors list, the pattern rotation must be repeated three times to create enough patterns for the ten regions.

```
pattern1 value=m3n0 r=3;
```

Define a solid pattern for the blocks. PATTERN2 through PATTERN6 define the patterns for the block surfaces.

```
pattern2 value=solid color=green;
pattern3 value=solid color=cyan;
pattern4 value=solid color=lime;
pattern5 value=solid color=blue;
pattern6 value=solid color=red;
```

Define legend characteristics. ACROSS= places all the legend values on one line.

```
legend1 shape=bar(2,4) across=5
        value=(j=l)
        label=('Number' j=l 'of Sites:')
        frame;
```

Produce the block maps. The FORMAT statement assigns SITESFMT. to the response variable. DISCRETE specifies that each formatted value is a separate response level. AREA= specifies that the map surface should be patterned by the first variable in the ID statement, REGION. CBLKOUT= and COUTLINE= specify the color that outlines the blocks and the regions, respectively.

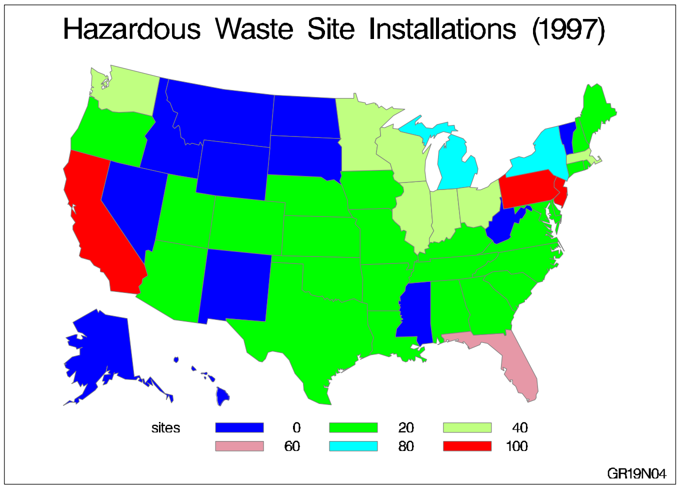
```
proc gmap map=reflib.states2 data=reflib.sites;
   format sites sitesfmt.;
   id region state;
   block sites / discrete
                 area=1
                 legend=legend1
                 shape=block
                 cblkout=black
                 coutline=black;
run;
quit;
```

Example 4: Producing a Simple Choropleth Map

Procedure features:
ID statement
CHORO statement option:
    COUTLINE=

Data set: REFLIB.SITES

Sample library member: GR19N04



This example produces a choropleth (2D) map that shows the total number of hazardous waste sites in each state in 1997. Since the DISCRETE option is not used, the response variable is assumed to have a continuous range of values. Because neither the LEVELS= nor MIDPOINTS= option is used, the GMAP procedure selects a number of levels based on the number of map areas and then calculates the appropriate response levels. The legend shows the midpoint value of each level.

The map areas use the default pattern, which is a solid fill that rotates through the colors list. Because the colors list is specified in the GOPTIONS statement, all colors are used in the rotation. COUTLINE= outlines the map areas in gray, instead of the default outline color, which is the first color in the list, in this case, BLUE.

Assign the librefs and set the graphics environment. COLORS= specifies the colors list, which is used by the default patterns and outlines. CTEXT= specifies the color for all text on the output.

```
libname reflib 'SAS-data-library';
libname maps 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
         colors=(blue green lime lipk cyan red)
         ctext=black ftext=swiss htitle=6 htext=3;
```

Define title and footnote for map.

```
title1 'Hazardous Waste Site Installations (1997)';
footnote1 j=r 'GR19N04 ';
```

Produce the choropleth map. The ID statement specifies the variable that is in both the map data set and the response data set that defines map areas. COUTLINE= specifies the color for the map area outlines.

```
proc gmap map=maps.us data=reflib.sites;
   id state;
   choro sites / coutline=gray;
run;
quit;
```

Example 5: Creating Maps with Drill-down for the Web

Procedure Features:

CHORO statement options:

- DES=
- DISCRETE
- HTML=
- NAME=

BLOCK statement options:

- BLOCKSIZE=
- DES=
- MIDPOINTS=
- NAME=

ODS Features:

ODS HTML statement :

- BODY=

CONTENTS=
FRAME=
NOGTITLE
PATH=

Other Features:

BY statement
GOPTIONS statement
LEGEND statement
PATTERN statement
TITLE statement

Sample library member: GR19N05

This example shows how to create a 2D choropleth map with simple drill-down functionality for the Web. When this map is displayed in a browser, you can select an area of the map and display additional information about the data.

The example explains how to use the ODS HTML statement and the HTML procedure options to create the drill-down. It shows how to

- explicitly name the HTML files and open and close them throughout the program
- use BY-group processing with ODS HTML, including storing multiple graphs in one file and incrementing anchor names, catalog entry names, and graphics file names
- use the PATH= option to specify the destination for the HTML and GIF files created by the ODS HTML statement
- use the NAME= option to name the graphics catalog entries
- assign anchor names to the graphics output with the ANCHOR= option in the ODS HTML statement
- add an HTML HREF string to a data set to define a link target
- assign link targets with the HTML= procedure option
- use DES= to control the text of the table of contents entry
- suppress the titles in the GIF files and display them in the HTML file.

For more information, see “ODS HTML Statement” on page 200.

The example also illustrates other CHORO and BLOCK statement options.

The program produces one choro map that shows Environmental Protection Agency (EPA) regions and block maps of the states in each region. Each block map shows the number of hazardous waste sites for each state in the selected region. Figure 19.10 on page 781 shows the map of the EPA regions.

Figure 19.10 Browser View of Regional Map

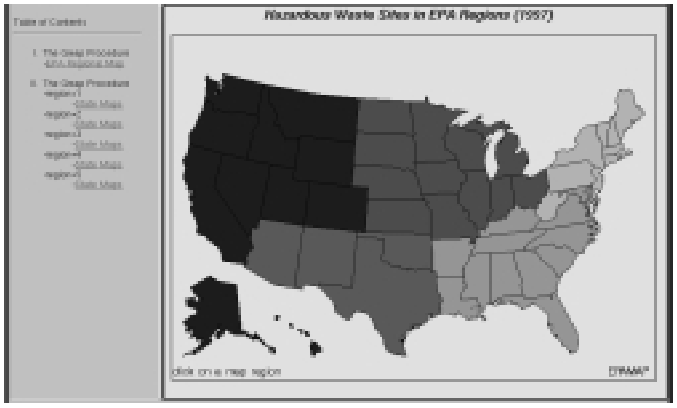
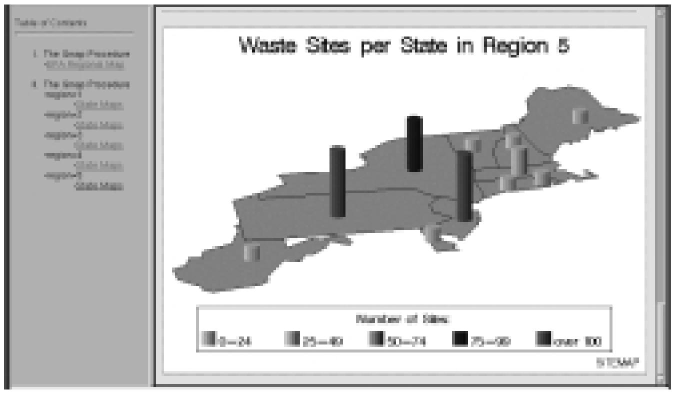


Figure 19.11 on page 781 shows the block map that appears when you select Region 5 in the map.

Figure 19.11 Browser View of Region 5 Block Map



Assign the libref and the Web-server path. FILENAME assigns the fileref ODSOUT, which specifies a destination for the HTML and GIF files produced by the example program. To assign that location as the HTML destination for program output, ODSOUT is specified later in the program on the ODS HTML statement's PATH= option. ODSOUT must point to a Web-server location if procedure output is to be viewed on the Web.

```
libname maps 'SAS-MAPS-library';
filename odsout 'path-to-Web-server-space';
```

Close the ODS Listing destination for procedure output, and set the graphics environment. To conserve system resources, ODS LISTING closes the Listing destination for procedure output. Thus, the graphics output is not displayed in the GRAPH window, although it is written to the catalog.

```
ods listing close;
goptions reset=global gunit=pct cback=white
```

```
         colors=(black blue green red)
         ftext=swiss htitle=6 htext=3.5;
```

Create the data set STATES. STATES contains the FIPS codes for each state and the total number of hazardous waste sites installed in each state.

```
data sites;
   length stcode $ 2;
   input region stcode $ sites;
   state=stfips(stcode);
   datalines;
1   AK  12
4   AL  7
4   AR  12
2   AZ  10
1   CA  90
1   CO  15
5   CT  15
5   DE  18
4   FL  52
4   GA  15
1   HI  4
3   IA  16
1   ID  8
3   IL  38
3   IN  30
3   KS  10
4   KY  16
4   LA  15
5   MA  30
4   MD  13
5   ME  12
3   MI  72
3   MN  30
3   MO  22
4   MS  1
1   MT  8
4   NC  22
3   ND  0
3   NE  10
5   NH  18
5   NJ  105
2   NM  9
1   NV  1
5   NY  78
3   OH  34
2   OK  10
1   OR  10
5   PA  100
5   RI  12
4   SC  26
3   SD  2
4   TN  14
```

```
2   TX  26
1   UT  12
4   VA  25
5   VT  8
1   WA  49
3   WI  40
5   WV  6
1   WY  3
;
```

Add the HTML variable to SITES and create the NEWSITES data set. The HTML variable REGIONDRILL contains the targets for the values of the variable REGION.

```
data newsites;
   length regiondrill $40;
   set sites;
   if region=1 then
      regiondrill='HREF="hazsite_statebody.html#Region1"';
   if region=2 then
      regiondrill='HREF="hazsite_statebody.html#Region2"';
   if region=3 then
      regiondrill='HREF="hazsite_statebody.html#Region3"';
   if region=4 then
      regiondrill='HREF="hazsite_statebody.html#Region4"';
   if region=5 then
      regiondrill='HREF="hazsite_statebody.html#Region5"';
run;
```

Assign graphics options for producing the ODS HTML output. DEVICE=GIF causes the ODS HTML statement to generate the graphics output as GIF files. TRANSPARENCY causes the graphics output to use the Web-page background as the background of the graph.

```
goptions device=gif transparency;
```

Open the ODS HTML destination. BODY= names the file for storing HTML output. CONTENTS= names the HTML file that contains the table of contents to the HTML procedure output. The contents file links to each of the body files written to the HTML destination. FRAME= names the HTML file that integrates the contents and body files. NOGTITLE suppress the graph titles from the SAS/GRAPH output and displays them through the HTML page. PATH= specifies the ODSOUT fileref as the HTML destination for all the HTML and GIF files.

```
ods html body='hazsite_mapbody.html'
         contents='hazsite_contents.html'
         frame='hazsite_frame.html'
         nogtitle
         path=odsout;
```

Define the title and footnote for the map of the EPA regions.

```
title1 'Hazardous Waste Sites in EPA Regions (1997)';
footnote1 h=3 j=l 'click on a map region' j=r 'EPAMAP ';
```

Define a map pattern for each region. Each PATTERN statement defines one map/plot pattern. The patterns are assigned to the map areas that represent the EPA regions

```
pattern1 value=msolid color=blue;
pattern2 value=msolid color=green;
pattern3 value=msolid color=red;
pattern4 value=msolid color=lime;
pattern5 value=msolid color=cyan;
```

Generate the regional map. The ID statement specifies the variable that defines the map areas and is in both the map data set and the response data set. DISCRETE specifies that each value of the numeric response variable, STATE, be treated as a separate response level. HTML= specifies REGIONDRILL as the variable that contains the targets for the map regions. Specifying HTML variables causes SAS/GRAPH to add an image map to the HTML body file. DES= specifies the description that is stored in the catalog and used in the Table of Contents. NAME= specifies the name of the graphics catalog entry. Because the PATH= destination is a file storage location and not a specific file name, the catalog entry name EPAMAP is automatically assigned to the GIF file.

```
proc gmap map=maps.us data=newsites;
   id state;
   choro region / discrete
                  html=regiondrill
                  coutline=black
                  nolegend
                  des='EPA Regional Map'
                  name='epamap';
run;
quit;
```

Open a new body file for the state maps. Assigning a new body file closes HAZSITE_MAPBODY.HTML. The contents and frame files, which remain open, will provide links to all body files. ANCHOR= specifies the name of the anchor that identifies the link target. This name is automatically incremented when the graphics output is generated. GTITLE uses titles in the GIF files.

```
ods html body='hazsite_statebody.html'
         anchor='Region1'
         gtitle
         path=odsout;
```

Assign new graphics options for ODS HTML output.

```
goptions notransparency
         border;
```

Sort the response data set NEWSITES in order of the BY variable. The data must be in sorted order before running the GMAP procedure with BY-group processing.

```
proc sort data=newsites;
   by region;
run;
```

Define legend characteristics for the state maps. VALUE= specifies text for the legend values that describes the ranges specified by MIDPOINTS= in the BLOCK statement.

```
legend1 shape=bar(3,4)
        label=('Number of Sites'
               position=(top center))
        value=(j=l '0-24' '25-49' '50-74' '75-99' 'over 100')
        frame;
```

Define a pattern for the map areas. Because the procedure uses BY-group processing to generate the maps, all map areas use the same map pattern.

```
pattern1 value=ms color=gray;
```

Define the patterns for the blocks.

```
pattern2 value=solid color=lipk;
pattern3 value=solid color=cyan;
pattern4 value=solid color=green;
pattern5 value=solid color=blue;
pattern6 value=solid color=red;
```

Suppress the default BY-line and define a title that includes the BY-values. #BYVAL inserts the value of the BY variable into the title of each block map.

```
options nobyline;
title1 'Wastes Sites per State in Region #byval(region)';
footnote1 h=3 j=r 'SITEMAP ';
```

Generate the block maps for each region. MIDPOINTS= defines the midpoints of the ranges described in the legend. NAME= is a full 8 characters ending in **1** so the incremented names match the regions. NAME= specifies the name of the first catalog entry. Because BY-group processing generates multiple graphs from one BLOCK statement, the name assigned by NAME= is incremented to provide a unique name for each piece of output. These names are automatically assigned to the GIF files. DES= specifies the description that is stored in the catalog and used in the Table of Contents. Because BY-group processing is used, the same description is assigned to all the output.

```
proc gmap map=maps.us data=newsites;
   by region;
   id state;
   block sites / midpoints=(12 37 62 87 112)
```

```
                     legend=legend1
                     shape=cylinder
                     blocksize=4
                     coutline=black
                     des='State Maps'
                     name='states01';
run;
quit;
```

Close the ODS HTML destination, and open the ODS Listing destination. You must close the HTML destination before you can view the output with a browser.

```
ods html close;
ods listing;
```

Example 6: Labeling the States on a U.S. Map

Procedure features:

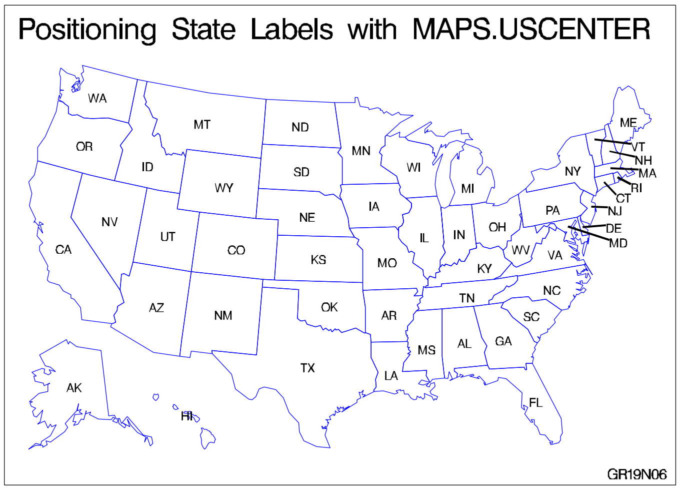
CHORO statement options:

ANNOTATE=
NOLEGEND

Other features:

Annotate Facility

Sample library member: GR19N06



This example uses the MAPS.USCENTER data set and the Annotate facility to add postal code labels to each state. The program first builds an Annotate data set that contains the instructions for drawing the labels. Some of the labels are in the center of the state and others use external labeling with leader lines. The CHORO statement assigns the Annotate data set to the map.

Note: The coordinates in MAPS.USCENTER have been projected to match coordinates in the MAPS.US data set. △

Assign the librefs and set the graphics environment.

```
libname reflib 'SAS-data-library';
libname maps 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
         colors=(black blue green red)
         ftext=swiss htitle=6 htext=3;
```

Create annotate data set, REFLIB.CENTER, from MAPS.USCENTER. The annotate data set labels each state with a two-letter abbreviation. MAPS.USCENTER provides the x and y coordinates for the labels. FLAG, which is initially turned off, signals when external labeling is in effect. The labels are drawn after the map because the value of WHEN is **a** (after).

```
data reflib.center;
   length function $ 8;
   retain flag 0 xsys ysys '2' hsys '3' when 'a'
          style 'swiss';
   set maps.uscenter
       (where=(fipstate(state) ne 'DC')
       drop=long lat);
```

The FIPSTATE function creates the label text by converting the FIPS codes from MAPS.USCENTER to two-letter postal codes.

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

If the labeling coordinates are outside the state (**OCEAN='Y'**), Annotate adds the label and prepares to draw the leader line. **Note:** OCEAN is a character variable and is, therefore, case sensitive. **OCEAN='Y'** must specify an uppercase Y.

```
   if ocean='Y' then
      do;
         position='6';
         output;
         function='move';
         flag=1;
      end;
```

When external labeling is in effect, Annotate draws the leader line and resets the flag.

```
   else if flag=1 then
      do;
```

```
            function='draw';
            size=.25;
            flag=0;
         end;
      output;
   run;
```

Define title and footnote for map.

```
title 'Positioning State Labels with MAPS.USCENTER';
footnote j=r 'GR19N06 ';
```

Define pattern characteristics. PATTERN1 defines a single map pattern that is repeated for each of the 50 map areas (states). The pattern is an empty fill with a blue border. VALUE= defines a map/plot pattern, which cannot be used by the blocks. Specifying a color causes PATTERN1 to generate only one pattern definition. REPEAT= specifies the number of times to repeat the pattern definition.

```
pattern1 value=mempty color=blue repeat=50;
```

Produce the choropleth map. NOLEGEND suppresses the legend. ANNOTATE= specifies the data set to annotate the map.

```
proc gmap data=maps.us map=maps.us;
   id state;
   choro state / nolegend
                 annotate=reflib.center;
run;
quit;
```

Example 7: Producing a Simple Prism Map

Procedure features:

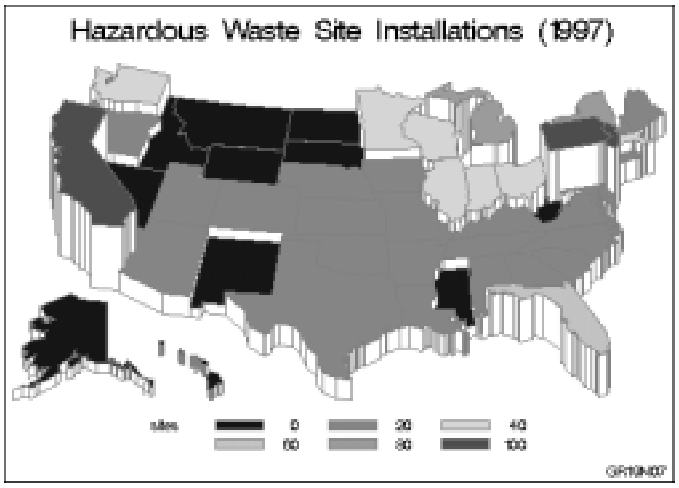
ID statement

PRISM statement option:

COUTLINE=

Data set: REFLIB.SITES

Sample library member: GR19N07



This example produces a prism map of the hazardous waste sites. Since the DISCRETE option is not used, the response variable is assumed to have a continuous range of values. Because neither the LEVELS= nor MIDPOINTS= option is used, the GMAP procedure selects a number of levels based on the number of map areas and then calculates the appropriate response levels. The legend shows the midpoint value of each level.

The map areas use the default pattern, which is a solid fill that rotates through the colors list. Because the colors list is specified in the GOPTIONS statement, all colors are used in the rotation. COUTLINE= outlines the map areas in gray, instead of the default outline color, which is the first color in the list, in this case, BLUE.

Since the XVIEW=, YVIEW=, and ZVIEW= options are not used, the default viewing position, above and to the east and south of the center of the map, is used. Since the XLIGHT= and YLIGHT= options are not used, none of the side polygons of the prisms are shadowed. The light source is the same as the viewing position.

Assign the librefs and set the graphics environment. COLORS= specifies the colors list, which is used by the default patterns and outlines. CTEXT= specifies the color for all text.

```
libname reflib 'SAS-data-library';
libname maps 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
         colors=(blue green lime lipk cyan red)
         ctext=black ftext=swiss htitle=6 htext=3;
```

Define title and footnote for the map.

```
title1 'Hazardous Waste Site Installations (1997)';
footnote1 j=r 'GR19N07 ';
```

Produce the prism map. The ID statement specifies the variable in the map data set and the response data set that defines map areas. COUTLINE= specifies the map area outline color.

```
proc gmap map=maps.us data=reflib.sites;
   id state;
   prism sites / coutline=gray;
run;
```

```
quit;
```

Example 8: Specifying Midpoints in a Prism Map

Procedure features:

PRISM statement options:

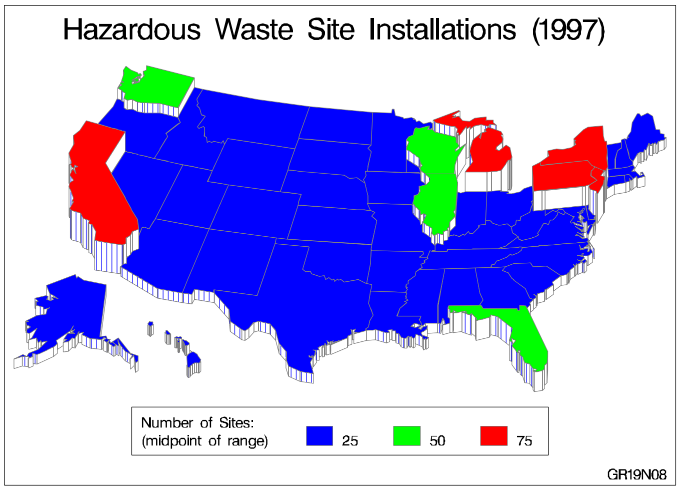
- LEGEND=
- MIDPOINTS=
- XLIGHT=
- XVIEW=
- ZVIEW=

Other features:

LEGEND statement

Data set: REFLIB.SITES

Sample library member: GR19N08



This example explicitly specifies the midpoints for three response levels. Each response level uses the default solid pattern and a color from the colors list.

The example also changes the map viewpoint and light source.

Assign the librefs and set the graphics environment. COLORS= specifies the colors list, which is used by the default patterns and outlines. CTEXT= specifies the color for all text.

```
libname reflib 'SAS-data-library';
libname maps 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
         colors=(blue green red) ctext=black
         ftext=swiss htitle=6 htext=3;
```

Define title and footnote for map.

```
title1 'Hazardous Waste Site Installations (1997)';
footnote1 j=r 'GR19N08 ';
```

Define legend characteristics. CBORDER= draws a black frame around the legend. If FRAME were specified, it would be BLUE, the first color in the colors list.

```
legend shape=bar(4,4)
       value=(j=l)
       label=('Number of Sites:'
              j=l '(midpoint of range)')
       cborder=black;
```

Produce the prism map. MIDPOINTS= specifies three response levels for the map. XLIGHT= moves the light source to the right and adds shadows to the left-side polygons of the prisms. XVIEW= and ZVIEW= shift the viewing point to the right and upward, respectively. This reduces the number of prisms that are partially hidden by taller neighbors.

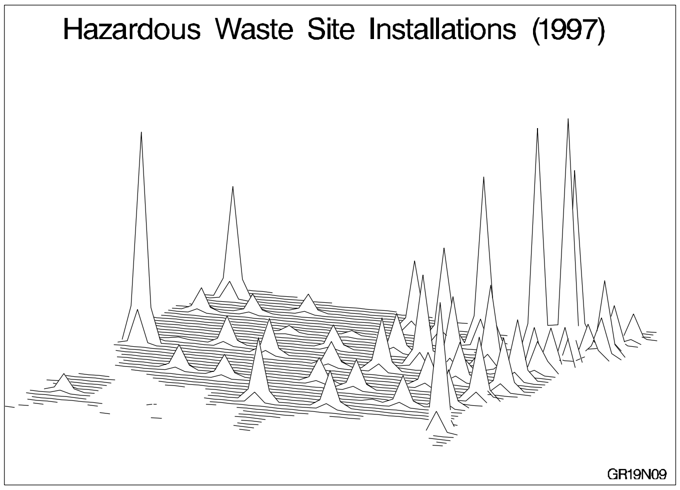
```
proc gmap map=maps.us data=reflib.sites;
   id state;
   prism sites / midpoints=25 50 75
                 xlight=5
                 xview=.75
                 zview=5
                 legend=legend
                 coutline=gray;
run;
quit;
```

Example 9: Producing a Simple Surface Map

Procedure features:
SURFACE statement

Data set: REFLIB.SITES

Sample library member: GR19N09



This example produces a surface map that shows the total number of hazardous waste sites in each state in 1997. Because the CONSTANT= and NLINES= options are not used, the GMAP procedure draws a surface that consists of 50 lines and uses the default decay function to calculate spike height and base width. And because the ROTATE= and TILT= options are not used, the map is rotated 70 degrees around the Z axis and tilted 70 degrees with respect to the X axis.

Assign the librefs and set the graphics environment. COLORS= specifies the colors list. By default the map uses the first color in the list.

```
libname reflib 'SAS-data-library';
libname maps 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
         colors=(black blue green red)
         ftext=swiss htitle=6 htext=3;
```

Define title and footnote for the map.

```
title1 'Hazardous Waste Site Installations (1997)';
footnote1 j=r 'GR19N09 ';
```

Produce the surface map. The ID statement specifies the variable in the map data set and the response data set that defines the map areas.

```
proc gmap map=maps.us data=reflib.sites;
   id state;
   surface sites;
run;
quit;
```

Example 10: Rotating and Tilting a Surface Map

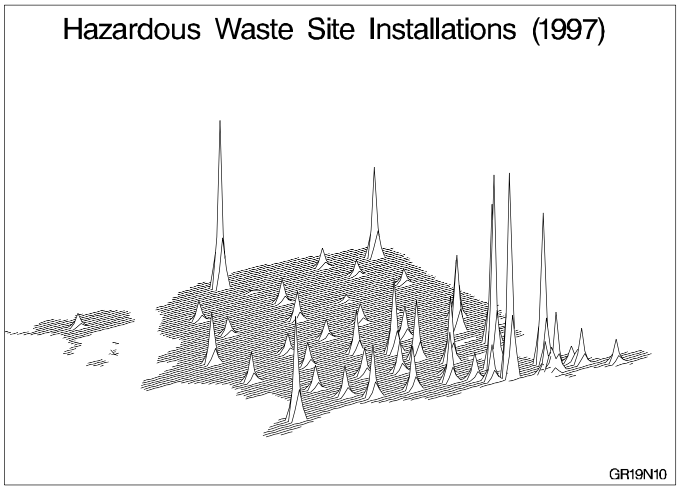
Procedure features:

SURFACE statement options:

CONSTANT=
NLINES=
ROTATE=
TILT=

Data set: REFLIB.SITES

Sample library member: GR19N10



This example tilts and rotates the surface map and uses more lines to draw the surface.

Assign the librefs and set the graphics environment.

```
libname reflib 'SAS-data-library';
libname maps 'SAS-data-library';
goptions reset=global gunit=pct border cback=white
         colors=(black blue green red)
         ftext=swiss htitle=6 htext=3;
```

Define title and footnote for the map.

```
title1 'Hazardous Waste Site Installations (1997)';
footnote1 j=r 'GR19N10 ';
```

Produce the surface map. CONSTANT= specifies a value that is less than the default value so the spikes are narrower at the base. NLINES= specifies the maximum number of map lines, which gives the best map shape resolution. ROTATE= and TILT= adjust the map orientation to make the crowded spikes in the northeast portion of the map easier to distinguish.

```
proc gmap map=maps.us data=reflib.sites;
   id state;
   surface sites / constant=4
```

```
                    nlines=100
                    rotate=40
                    tilt=60;
run;
quit;
```

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/GRAPH® Software: Reference, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS/GRAPH® Software: Reference, Version 8

ISBN 1-58025-525-6

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999