SAS/GRAPH® 9.4: Mapping Reference

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2016. *SAS/GRAPH® 9.4: Mapping Reference*. Cary, NC: SAS Institute Inc.

SAS/GRAPH® 9.4: Mapping Reference

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

January 2017

9.4-P2:grmapref

Contents

About This Book

Using This Document

Prerequisites

This document is written for users who are experienced in using both the SAS System and its graphics component, SAS/GRAPH. You should understand the concepts of programming in the SAS language. You should understand how to visually present data and output as graphics using SAS/GRAPH procedures. The following table summarizes the SAS System concepts that you need to understand in order to use SAS/GRAPH Mapping.

To learn how to	Refer to
invoke the SAS System at your site	instructions provided by the on-site SAS support personnel
use Base SAS software use the DATA step to create and manipulate SAS data sets use the SAS windowing environment or SAS Enterprise Guide to enter, edit, and submit program code	Base documentation library: • *SAS Language Reference: Concepts* • *SAS Data Set Options: Reference* • *SAS Formats and Informats: Reference* • *SAS Functions and CALL Routines: Reference* • *SAS Statements: Reference* • *SAS System Options: Reference* • *Base SAS Utilities: Reference*
allocate SAS libraries and assign librefs create external files and assign filerefs	documentation for using the SAS System in your operating environment: • *SAS Companion for Windows* • *SAS Companion for UNIX Environments* • *SAS Companion for z/OS*
manipulate SAS data sets using SAS procedures	*Base SAS Procedures Guide*
use device drivers to generate device-based graphics	*SAS/GRAPH: Reference*
enhance the appearance of your graphics output	*SAS/GRAPH: Mapping Reference*

Particular fonts have special meanings when used in the presentation of SAS/GRAPH syntax in this document. For example, items presented in italics identify arguments or values that you supply. Angle brackets (< >) indicate optional arguments. The conventions used in this document are the same conventions used in Base SAS documentation. For a complete explanation, see the Base SAS documentation listed above.

Map Data Sets

To draw maps, you need to know where the map data sets are stored on your system. Depending on your installation, the map data set might automatically be assigned a libref, such as MAPSGFK, MAPSSAS, or MAPS. The MAPSGFK map library contains vector-based maps data sets that SAS has licensed from GfK GeoMarketing GmbH. SAS updates this map library as it receives updates from GfK, and provides the library for use with SAS/GRAPH.Ask your on-site SAS support personnel or system administrator where the map data sets are stored for your site.

About Examples and Their Output

Most of the chapters in this document include examples that illustrate some of the features of a procedure or its statements. The output that is shown for the examples was generated in a Windows operating environment. If you are using a different operating environment, you might need to make some minor adjustments to the example programs.

In most cases, the output was sent to the HTML destination and generated using the default style and device for that destination. Exceptions are noted in the text.

The dimensions of the graphics output area vary across devices and when using the GRAPH windows. The dimensions can affect aspects of the graphics output – for example, the appearance of axes or the position of graphics elements that use explicit coordinates in units other than percent. You might need to adjust the dimensions of your graphics output area or the size of graphics elements to correct any differences that you see. Most of the images of output in this document were generated with a GOPTIONS statement. This statement generally specified a size approximately equal 5.5 inches by 4.2 inches. However, some images might be larger, if necessary, to accommodate the content of the graph.

```
goptions hsize=5.5in vsize=4.2in;
```

These HSIZE= and VSIZE= settings are not shown in the example code and are not necessary for generating the output. However, you might want to use similar settings if your output looks significantly different from the output that is shown in the document.

Most examples specify these options:

RESET=ALL
: sets all graphics options to default values and cancels all global statements.

BORDER
: draws a border around the graphics output area.

The way that output is presented on your device depends on the environment in which you are running SAS/GRAPH.

Many examples process sample data contained in a SAS data set stored in the SASHELP library. The documentation for each example provides the data set name used. To see a brief description of any SAS data set in the SASHELP library, as well as output

displaying the first five observations in each data set, please refer to SASHELP Data Sets.

What's New in SAS/GRAPH 9.4: Mapping Reference

Overview

There are many changes and enhancements for SAS/GRAPH 9.4. Highlights include the following:

- Many procedures have significant enhancements and new options. See "Mapping Procedures" on page xi for a complete list.
- The map data sets in the MAPSGFK library have been updated.
- For ease of use, the mapping functions are removed from the SAS/GRAPH: Reference, Fifth Edition, and placed in a new document, SAS/GRAPH: Mapping Reference. This document includes the GEOCODE, GINSIDE, GMAP, GPROJECT, GREDUCE, GREMOVE, and MAPIMPORT procedures, and their supporting documentation.
- TITLE and FOOTNOTE statements now support the ALT=*"text-string"* option. See "Global Statements" in *SAS/GRAPH: Reference*.
- A topic in the introduction consolidates links for examples and various resources on the SAS website.
- The sample program GEOSTRT.SAS is updated to use the TYPE= option and create a custom GCTYPE lookup data set that includes an uncommon abbreviation for Boulevard.
- Starting with the fourth maintenance release for SAS 9.4, information about creating maps was moved from the SAS/GRAPH Reference to this separate document, *SAS/GRAPH: Mapping Reference*. This document includes the GEOCODE, GINSIDE, GMAP, GPROJECT, GREDUCE, GREMOVE, and MAPIMPORT procedures, and their supporting documentation.

Mapping Procedures

GEOCODE Procedure

The **GEOCODE procedure** has the following new functions, options, and enhancements:

- The GEOCODE procedure now supports street geocoding for Canadian provinces. As a result, new lookup data sets are available, and the existing lookup data sets are reformatted. The format changes include the replacement of FIPS codes with state and city names represented by character variables. The new format data sets must be processed with the SAS 9.4 version of the geocoder. A SAS geocoder from an earlier release must process lookup data sets with a format corresponding to that release of SAS.
- SAS makes available a new import macro program that creates Canadian street lookup data sets in the new format only. %GEOBASE2GEOCODE imports Canadian roadways and addresses, creating the lookup data sets used by the GEOCODE procedure's STREET geocoding method. This macro program can be downloaded from the SAS Maps Online web site.
- The %TIGER2GEOCODE import macro program is updated to create U.S. lookup data sets in the new format. This format is used by the GEOCODE procedure's STREET geocoding method. This macro program enables you to import the U.S. Census Bureau's TIGER shapefiles for specific states and counties from the year 2007 or later. This new version as well as older versions of the %TIGER2GEOCODE macro program can be downloaded from the SAS Maps Online web site. The lookup data sets created from these older macro versions are in the format used only by releases prior to SAS 9.4.
- In support of Canadian street geocoding, the new DIRECTION=<*data-set*> option enables you to specify an alternate data set of street direction names and abbreviations, such as northwest or NW. The default data set is **SASHELP.GCDIRECT**, and it can be updated at any time. The SAS/GRAPH 9.4 version of this data set contains U.S. street directional text strings. The %GEOBASE2GEOCODE macro program creates a version of this lookup data set named **GCDIRECT_CAN** for use in Canadian street geocoding.
- In support of Canadian street geocoding, the new STATE=<*data-set*> option enables you to specify an alternate data set that contains state or province character names rather than FIPS numeric codes. The default data set is **SASHELP.GCSTATE**, and is installed with SAS/GRAPH 9.4. It can be updated at any time. This data set contains U.S. states and its overseas territories, and all Canadian provinces. As street geocoding data for other nations is acquired, you can update this data set to include additional state or region names with their abbreviations.
- In support of Canadian geocoding, a new street-type data set for Canadian roadways and addresses is available. The **SASHELP.GCTYPE_CAN** data set is created by the %GEOBASE2GEOCODE import program, and it can be updated at any time. Specify that the geocoder use this alternate data set with the TYPE= option when geocoding streets in Canada.
- The geocoder now supports street type prefixes as well as suffixes. The STREET geocoding method uses the street type data set to convert street type suffixes and prefixes from the input address observation to standardized forms. Prefixes are primarily used in non-U.S. street geocoding. An example is the Canadian address Boulevard Quebec, found in the **SASHELP.GCTYPE_CAN** data set. This data set is created by the %GEOBASE2GEOCODE import program. To accommodate both prefixes and suffixes, the format of the street type data set differs from that of previous SAS releases. The default version of this data set is **SASHELP.GCTYPE**. It contains standard U.S. street types and it is installed with SAS/GRAPH.
- Several of the data sets used in geocoding are indexed. These data sets need to be moved or copied with the DATASETS or COPY procedure to avoid losing the index file. Do not use local operating system utilities. A new check for these indexes was

added to the GEOCODE procedure in SAS 9.4. A warning is printed to the SAS log if the index is missing.

- Because the FIPS codes are replaced with variables containing the state and city names in character format, the **SASHELP.PLFIPS** lookup data set is no longer accessed by the geocoder. It is still installed with SAS/GRAPH because the data set is accessed by other applications.
- The STREET geocoding method, in supporting U.S. and Canadian addresses, now provides a "State mismatch" value for the output variable _MATCHED_ . This value indicates that a match was found for the city and country, but the state or province value was not matched.
- The STREET geocoding method, in supporting U.S. and Canadian addresses, now provides several new and changed values for the output variable _NOTES_. The value NOCT was deleted because the FIPS lookup data set is no longer used by street geocoding. The NS and NSM values were added to cover state matches and no matches. With the added support for street type prefixes, the value NOTYA was replaced with NOTPA and NOTSA and the value NOTYM was replaced with NOTPM and NOTSM. The value MZS was replaced with MCS to cover the cases when multiple matches are found for street and city and state.
- The sample program GEOSTRT.SAS is updated so that the TRACTCE00 variable is no longer requested to be written to the output data set. The variable no longer exists in the lookup data set **SASHELP.GEOEXM**.
- Starting with the first maintenance release for SAS 9.4, the GEOCODE procedure supports the following street and city geocoding new enhancements and clarifications:
  - The GROUP variable is added to the SASHELP.GCTYPE and SASHELP.GCTYPE_CAN lookup data sets that contain street type abbreviations for U.S. and Canada geocoding, respectively. This variable contains multiple abbreviations for the same street type. For example, AVENUE is abbreviated as AVE in English-speaking areas and as AV in certain provinces in Canada.
  - An index is no longer provided or required for the SASHELP.GCTYPE data set. SAS supplies indexes with the SASHELP.GCSTATE lookup data set, and the GEOCODE procedure looks for them when performing street geocoding. If you change the location of the GCSTATE data set, you must also move its associated data set index. If you create a customized version of this data set, you must create the data set indexes as well.
  - Each of the DIRABRV and DIRECTION variables in the SASHELP.GCDIRECT data set is capable of containing text strings. However, the DIRECTION variable should contain alphabetic characters only.
  - Three new note value tokens were added. These denote instances where the street geocoder detected different nonmissing city, state, or ZIP code values between the lookup data set and the input address data. The token for a no city match is NOCTM. The token for a no state match is NOSTM. The token for a no ZIP code match is NOZCM. The token strings are contained in the _NOTES_ variable that the street geocoder creates in the output data set.

    The missing value point (MVP) note token was added to denote instances where the street geocoder detected that a user-supplied street lookup data set had missing X or Y coordinates.
  - Two variables now support city geocoding with the SASHELP.ZIPCODE lookup data set. In this data set the CITY2 and STATENAME2 variables are both normalized—containing only uppercase alphanumeric characters.

- Starting with the second maintenance release for SAS 9.4, the GEOCODE procedure supports the following enhancements and changes:
  - The U.S. CITY geocoding method supports non-abbreviated state names, just as the STREET geocoding method does. As a result of this enhancement, customized versions of the SASHELP.GCSTATE lookup data set no longer need to maintain the original data set's sort order. This data set is used by the CITY method geocoder in determining state name matches. Also, in a customized version of this street state data set, duplicate values are allowed in the variable containing the state or province postal service abbreviations. The geocoder uses these values as the key to grouping equivalent state names. Make sure that you keep the values unique. For example, avoid using an 'MI' abbreviation for both Mississippi and Michigan.
  - The geocoder supports optional nonstandard state values; expanding the existing support of two-character postal codes and the non-abbreviated state names. The nonstandard state values are contained in a new variable named StateAlias within the SASHELP.GCSTATE lookup data set. Examples of nonstandard state names or IDs for North Carolina are 'N. Car.' or 'No. Car.'. Multiple values for one state must be separated by a single vertical bar. These nonstandard variable values can be in a customized version of SASHELP.GCSTATE lookup data set, or in the input address data set. Traditional Canadian postal abbreviations for provinces and territories are included in this update.
  - The values of the variables MAPIDNAME (state or province name), and MAPIDNAME2 (normalized state or province name) are changed. These variables are in the SASHELP.GCSTATE lookup data set. The variable values now match, respectively, the ISONAME and ISONAME2 variables found in the MAPSGFK.WORLD_CITIES data set.
  - An improved %TIGER2GEOCODE import program was used to update street lookup data sets (USM, USS, and USP). This provides the geocoder the opportunity to find better street matches for any given TIGER release from 2009–2013. Download these updated street lookup data sets from the SAS Maps Online web site. The previous data sets were renamed and all are differentiated by version. You can use only the lookup data sets that correspond to your SAS release. The updated TIGER2GEOCODE.sas program is also available for download from SAS Maps Online. This program converts TIGER/Line shapefiles into PROC GEOCODE street method lookup data sets, and this improved version provides more street matches for any specific SAS release.
- Starting with the third maintenance release for SAS 9.4, support is added for range geocoding with IPv6 addresses. A new version of the %MAXMIND autocall macro converts IPv6 geocoding data from MaxMind, Inc. into SAS data sets. Comma-separated-value (CSV) formatted files are available as downloads from MaxMind.
- Starting with the fourth maintenance release for SAS 9.4, the GEOCODE procedure supports the following street geocoding enhancements:
  - Street geocoding now obtains more accurate locations in areas where the U.S. Postal Service has reassigned local ZIP codes when modifying its delivery routes.
  - The sample program GEOSTRT.SAS is updated to use the TYPE= option and create a custom GCTYPE lookup data set that includes an uncommon abbreviation for Boulevard.

GINSIDE Procedure

The **GINSIDE procedure** has the following changes and enhancements:

- The sample programs **GINSIDE.SAS** and **GINSIDE2.SAS** are rewritten for clarity and standardization.
- Two new options are available to control whether to keep or drop all map data set variables before they are written to the output data set. Although this is the default behavior, you can specify KEEPMAPVARS in the GINSIDE statement to keep all map data set variables in the output data set. Conversely, specify DROPMAPVARS to keep the ID variable but drop all other map data set variables from the output data set.
- A section documenting how to optimize performance is now available in the GINSIDE procedure.

GMAP Procedure

The **GMAP procedure** has the following changes, enhancements, and new options:

- MAPSGFK map data sets are updated.
- The new LATLON option specifies that the unprojected LAT and LONG variables from the map data set are used for coordinate data instead of the Y and X variables. When the LATLON option is specified, the Y and X variables are no longer required by the GMAP procedure statement.
- The new sample program GMPUSDAT.SAS shows how to create a choropleth map using the sample response data set SASHELP.US_DATA provided with SAS/GRAPH 9.4.
- The new RESOLUTION= option specifies that the GMAP procedure use those map observations containing a resolution variable with a certain level (value). There are 10 resolution values that specify the screen resolution at which to display a map point. Setting this option to AUTO defaults to the resolution setting of the device being used in the GMAP procedure. RESOLUTION= NONE indicates that the DENSITY option, if specified, is used instead.
- The sample program GMPSIMPL.SAS is updated to use the new RESOLUTION= option instead of referring to the resolution variable directly.
- Starting with the first maintenance release for SAS 9.4, the CHORO statement in the GMAP procedure supports a production level of the OSM (OpenStreetMap) option. Use this option when displaying maps using a JAVA or JAVAIMG device. This is an appearance option that enables you to use the OpenStreetMap (OSM) map as a background map. You can specify no suboptions, use either a STYLE= suboption or an AUTOPROJECT suboption, or use both suboptions. If you specify the OSM option without any suboptions, the GMAP procedure by default uses the SASMAPNIK style and does not project the map data. When you specify the STYLE=osmstyle suboption, the GMAP procedure uses one of the supported OSM styles that are appropriate for the map that you are processing. Specifying the AUTOPROJECT suboption causes the GMAP procedure to project the map data from latitude and longitude coordinates (in degrees) onto the OpenStreetMap (OSM) map.

- An important distinction to note is that in the MAPSGfK map data sets, the positive values of the LONG variable (eastlong) go to the east from the prime meridian. The opposite is true in the traditional map data sets; they use westlong.

GPROJECT Procedure

The **GPROJECT procedure** has the following changes and enhancements:

- The LATLON option specifies that the LAT and LONG variables from the map data set are used for coordinate data instead of the Y and X variables. Now when the LATLON option is specified, the Y and X variables are no longer required by the PROC GPROJECT statement.
- The GPROJECT procedure can perform projections between any number of different projection types using the proj.4 system of projection strings. To do this, specify proj.4 on the PROJECT= option for the PROC GPROJECT statement. Proj.4 projection enables the transformation of geographic coordinates from either one projection or datum to another. Specifying the proj.4 projection with the GPROJECT procedure, by default, enables a transformation from latitude and longitude geographic coordinates (EPSG:4326) to an OpenStreetMap (OSM) coordinate system. Use the FROM= or TO= options to override either of these defaults. In the PROC GPROJECT statement, both the DEGREES option and the EASTLONG option are used by default with the proj.4 projection method.
- The FROM= option is available for the PROC GPROJECT statement. By specifying the FROM= option, you are automatically invoking the proj.4 projection system, and you are indicating a different coordinate system from which to start the transforming projection. You are overriding the use of the default latitude and longitude geographic coordinates (EPSG:4326).
- The TO= option is available for the PROC GPROJECT statement. By specifying the TO= option, you are automatically invoking the proj.4 projection system, and you are indicating a different coordinate system for the result of the transforming projection. You are overriding the use of the default OpenStreetMap (OSM) coordinate system. This system is also known as the Mercator or 900913 coordinate system.
- The FROM= and TO= options can also be used to reverse a projection. For example, if you already have an OSM projection, you can use the FROM= option in conjunction with the TO= option to revert the projection to EPSG:4326.

GREDUCE Procedure

The **GREDUCE procedure** has a new option. The LATLON option specifies that the unprojected LAT and LONG variables from the map data set are used for coordinate data instead of the Y and X variables. When the LATLON option is specified, the Y and X variables are no longer required by the GREDUCE procedure statement.

GREMOVE Procedure

The **GREMOVE procedure** has a new option. By default, all of the input data set variables are written to the output map data set. The DROPVARS option overrides this default behavior and omits the variables from the output map data set.

Global Statements Pertaining to Mapping

The global statements documented in this book pertain to controlling the appearance of output from the mapping procedures.

TITLE and FOOTNOTE statements now support the ALT=*"text-string"* option. This option enables you to specify descriptive text for the title or footnote. If you use ALT= in conjunction with the LINK= option, you can specify descriptive text for the URL to which the title or footnote links. The *"text-string"* can also contain occurrences of the variables named in a BY statement.

Starting with the third maintenance release for SAS 9.4, the new GraphTitle1Text style element is introduced. It controls and reduces the font size of the output of a TITLE1 statement in order to scale better with the graphs.

Part 1

SAS/GRAPH Mapping Concepts

Chapter 1

Introduction to SAS/GRAPH Mapping

Overview

SAS/GRAPH is a data visualization and presentation (graphics) component of the SAS System. As such, SAS/GRAPH does the following:

- organizes the presentation of your data and visually represents the relationship between data values as two- and three-dimensional graphs, including charts, plots, and maps.
- enhances the appearance of your output by allowing you to select text fonts, colors, patterns, and line styles, and control the size and position of many graphics elements.
- creates presentation graphics. SAS/GRAPH can create text slides, display several graphs at one time, combine graphs and text in one display, and create automated presentations.
- generates a variety of graphics output that you can display on your screen or in a web browser, store in catalogs, or review. You can send the graphics output to a hard copy graphics output device such as a laser printer.
- provides utility procedures and statements to manage the output.

This document specifically describes how to produce maps. The following topics describe the graphs that are produced by SAS/GRAPH mapping procedures and explain some of the features of SAS/GRAPH programs.

Components of SAS/GRAPH Mapping Software

There are several components to SAS/GRAPH software, but these two relate to creating maps.

SAS/GRAPH mapping procedures
: enable you to create a variety of maps. The SAS/GRAPH mapping procedures include GEOCODE, GINSIDE, GMAP, GPROJECT, GREDUCE, GREMOVE, and MAPIMPORT. With the exception of GMAP, these procedures are used to create or change map data. The GMAP procedure uses device drivers to generate visual output. SAS/GRAPH device drivers enable you to send output directly to your output device. Device drivers enable you to create output in a variety of formats such as PNG files and interactive ActiveX controls. This document, *SAS/GRAPH: Mapping Reference*, describes these procedures. The *SAS/GRAPH: Reference* describes how to use devices.

The Annotate Facility
: enables you to generate a special data set of graphics commands from which you can produce graphics output. This data set is referred to as an Annotate data set. You can use it to generate custom graphics or to enhance graphics output from many device-based SAS/GRAPH procedures, including GMAP. The *SAS/GRAPH: Reference* document describes this facility.

Producing Maps

SAS/GRAPH uses the GMAP procedure to produce two- and three-dimensional maps that can show an area or represent values of response variables for subareas. SAS/GRAPH includes data sets to produce geographic maps. In addition, you can create your own map data sets or import maps from Esri shapefiles.

Overview of Mapping

The GMAP procedure produces maps that display values or attributes related to areas on the map. The procedure can produce the following types of maps:

- block on page 5
- choropleth on page 5
- prism on page 6
- surface on page 6

The procedure can summarize data that vary by physical area, show trends and variations of data between geographic areas, and highlight regional differences or extremes.

SAS/GRAPH also provides several utility procedures for processing and handling map data. For example, you can convert map coordinates from spherical (longitude and

latitude) to a flat plane. For more information, see "Procedures for Handling Map Data" on page 7.

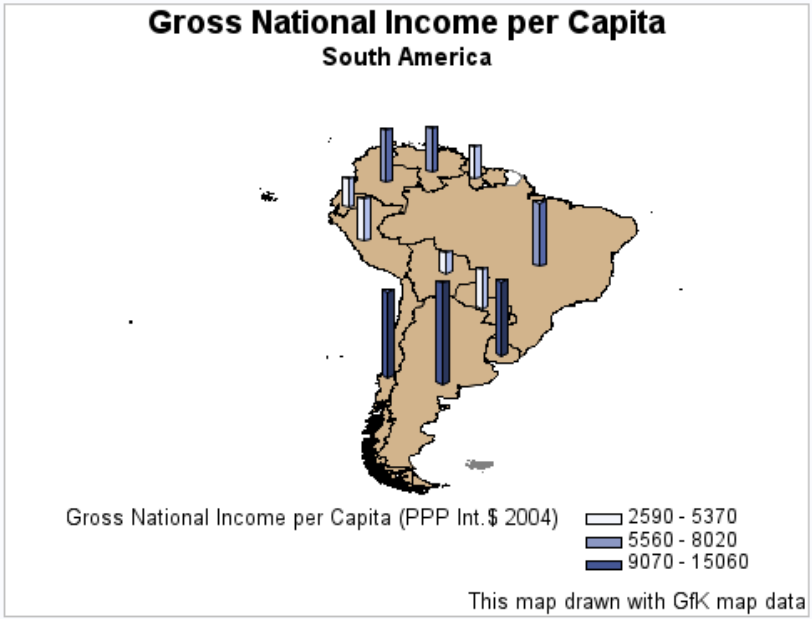
Options for controlling the appearance of maps include specifying the boundary density and controlling numeric ranges and categories. You can specify patterns, legends, and ODS styles. You can also use the SAS/GRAPH annotation feature to annotate graphs.

For more information and procedure syntax, see Chapter 7, "GMAP Procedure," on page 189.

Block Maps

Block maps are three-dimensional maps that represent data values as blocks of varying height rising from the middle of the map areas.

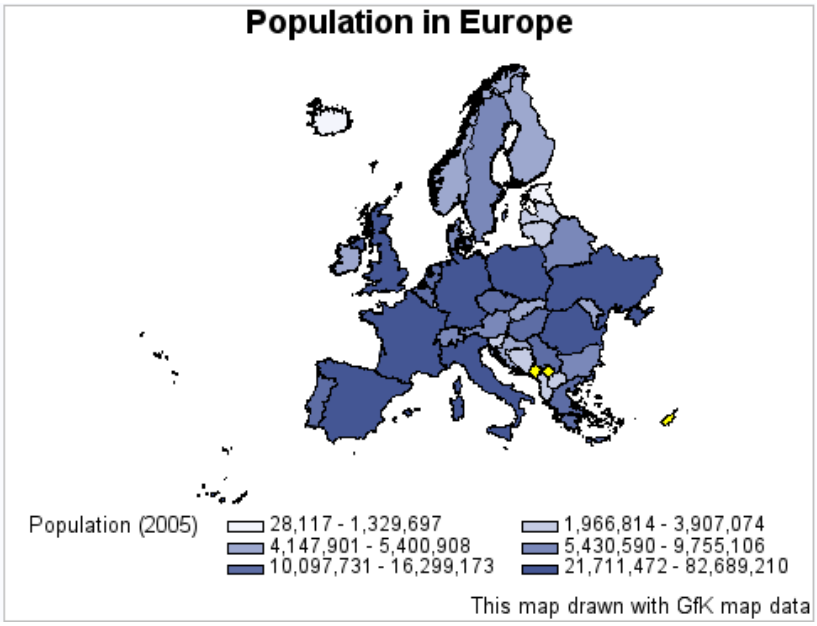
See Chapter 7, "GMAP Procedure," on page 189 for a complete description.



Choropleth Maps

Choropleth maps are two-dimensional maps that display data values by filling map areas with combinations of patterns and color that represent the data values. They are among the most commonly used and versatile map types.

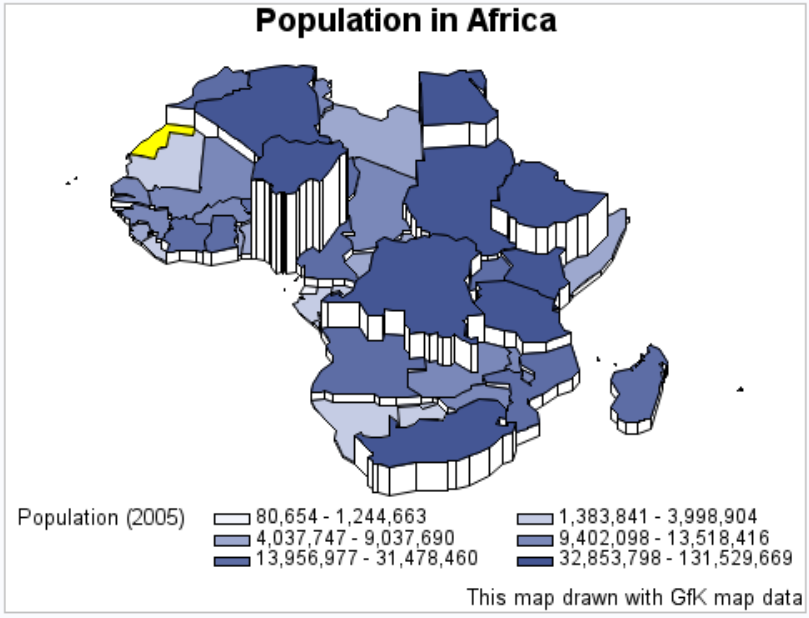
See Chapter 7, "GMAP Procedure," on page 189 for a complete description.



Prism Maps

Prism maps are three-dimensional maps that display data by raising the map areas and filling them with combinations of patterns and colors.

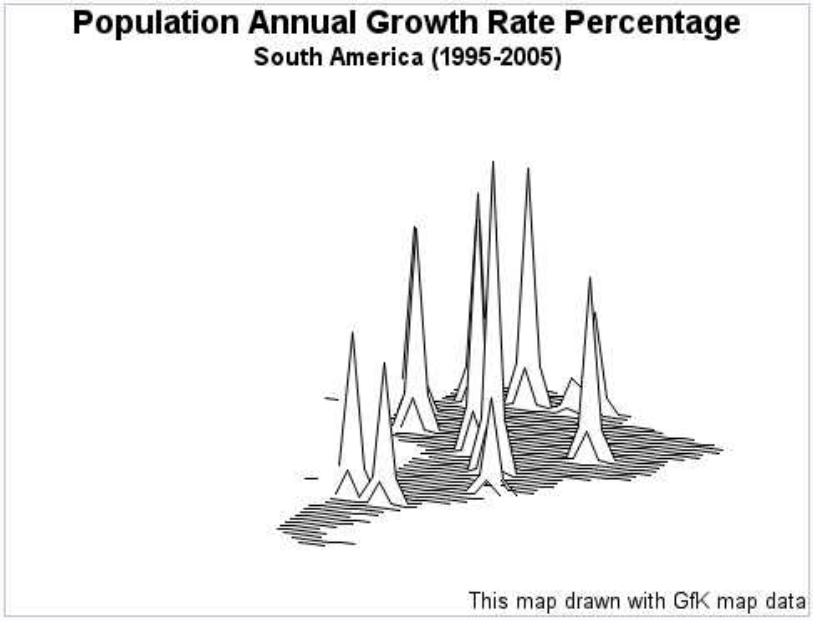
See Chapter 7, “GMAP Procedure,” on page 189 for a complete description.



Surface Maps

Surface maps are three-dimensional maps that represent data values as spikes of varying heights.

See Chapter 7, “GMAP Procedure,” on page 189 for a complete description.



Procedures for Handling Map Data

SAS/GRAPH provides several utility procedures for handling map data.

Procedure	Description
GEOCODE	Adds geographic coordinates to data sets that contain information such as street addresses. The coordinates typically represent the center of a ZIP code, a city, an address, or any geographic region. After geocoding, the coordinates can be used to display a point on a map or to calculate distances. Geocoding also enables you to add attributes values such as census blocks to an address. You can also use geocoding to associate ranges of IP addresses with locations.
GINSIDE	Compares a data set of X and Y coordinates to a map data set containing map polygons. The procedure determines whether the X and Y coordinates for each point fall inside or outside the map polygons. In addition, the procedure indicates the specific polygon that contains each inside point.
GPROJECT	Enables you to choose how geographic maps are projected. This is particularly important for large areas. Producing a map of any large area on the Earth involves distorting some areas in the process of projecting the spherical surface of the Earth onto a flat plane. You can use the procedure to select the projection method that least distorts your map.In addition, you can choose not to display a particular rectangular area of the map.
GREDUCE	Enables you to reduce the number of points in the data set. This is useful for large maps, where the amount of data can be prohibitively expensive in terms of computing resources or time to process.

Procedure	Description
GREMOVE	Enables you to remove boundary lines within a map. For example, if your map shows sales regions, you might want to remove state borders and show only the sales regions. The results of this procedure are used as the input map data set for the GMAP procedure.
MAPIMPORT	Converts Esri shapefiles into SAS map data sets.

Enhancing Graphics Mapping Output

SAS/GRAPH Statements and Global Options

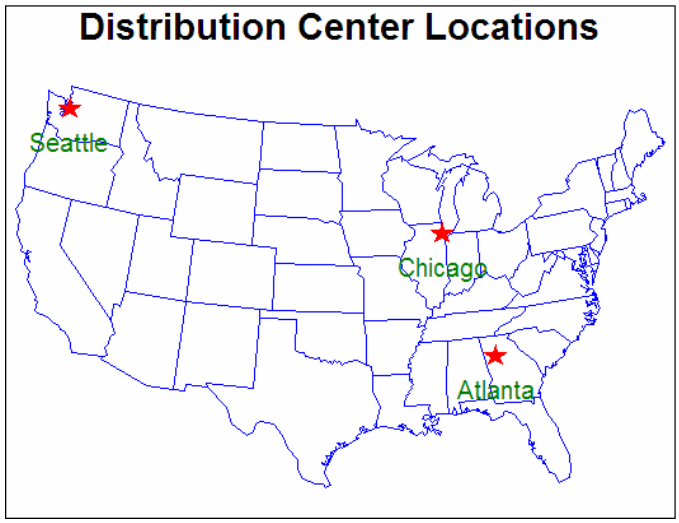
You can also use *global statements* and *graphics options* in SAS/GRAPH programs. With global statements, you can add titles and footnotes and control the appearance of axes, symbols, patterns, and legends. With graphics options, you can control the appearance of graphics elements by specifying default colors, fill patterns, fonts, text height, and so on. See "Enhancing Your Graphs" in *SAS/GRAPH: Reference* for more information.

The Annotate Facility

The Annotate facility enables you to programmatically create graphics by using certain variables in SAS data sets. It is often used to add text or special elements to the graphics output of other SAS/GRAPH procedures, although it can also be used to construct custom graphics output. Text and graphics can be placed at coordinates derived from input data, as well as coordinates expressed as explicit locations on the display. See "Overview: The Annotate Facility" in *SAS/GRAPH: Reference* or "Using Annotate Data Sets" in *SAS/GRAPH: Reference* for detailed information.

Figure 1.1 on page 9 shows GMAP procedure output annotated with stars and labels at selected cities.

Figure 1.1 *Annotate Graphics Applied to a Map*



The program that creates this output is in "Example 26: Labeling Cities on a Map" on page 347.

See "Displaying Images Using Annotate" in *SAS/GRAPH: Reference* for an example of how to display an image onto a map.

About the Sample Library

The documentation for each procedure, for global statements, and for features such as the Annotate facility provide examples that demonstrate these features of SAS/GRAPH. You can copy the example code from the Help, from the SAS website or from the zipped file of samples on the SAS support website and paste it into the Program Editor in your SAS Studio session.

In the SAS windowing environment, many of these samples are also included in the SAS Sample Library. How you access the code in the sample library depends on how it is installed at your site.

- In most operating environments, you can access the sample code through SAS Help and Documentation. Select **Help** ⇨ **SAS Help and Documentation**. On the **Contents** tab, select **Learning to Use SAS** ⇨ **Sample SAS Programs** ⇨ **SAS/GRAPH Samples**
- In other operating environments, the SAS Sample Library might be installed in your file system. Ask your on-site SAS support personnel where it is located on your system.

The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

This document includes the name of each sample file that it references (for example, GEOSMPL). The naming convention for SAS/GRAPH samples is G*pcxxxxx*, where *pc* is the product code and *xxxxx* is an abbreviation of the example title. The product code can be a code for a procedure, a statement, or in the case of Java and ActiveX examples, WB for "web graphs." To illustrate, the code for an example in the GMAP procedure

chapter, "Example 14: Combining Traditional Map Data and Sample Response Data to Map U.S. Population Statistics" on page 314, is stored in sample member GMPUSDAT.

- In the SAS Sample Library, the sample programs are organized by product. Within each product category, most of the samples are sorted by procedure name. Thus, to access the code for an example in the GMAP procedure chapter, select **Learning to Use SAS ⇨ SAS/GRAPH Samples**, scroll to **GMAP Procedure**, and select **GMPUSDAT-Combining Traditional and Sample Map Data to Map U.S. Population Statistics**.
- In your file system, the files that contain the sample code have filenames that match the sample member names. For example, in a directory-based system, the code for sample member GMPUSDAT is located in a file named GMPUSDAT.SAS.

Note: For Java and ActiveX (web graph) samples, the naming convention is GWB*xxxxx*.

Table 1.1 *Product Codes for SAS/GRAPH Mapping Procedures*

Procedure	Code
geocode	GE
ginside	IN
gmap	MP
gproject	PJ
greduce	RD
gremove	RM

Table 1.2 *Product Codes for SAS/GRAPH Statements Used in Mapping*

Statement	Code
by	BY
footnote	FO
goptions	ON
legend	LG
note	NO
pattern	PN
title	TI

Examples and Resources on the Web

The SAS website contains a large number of examples that can help you visualize and code your graphs.

- The SAS Training Post is a blog that provides tutorials, tips, and practical information about SAS. Dr. Robert Allison frequently contributes to the blog. His posts focus on using SAS/GRAPH for data visualization.

  http://blogs.sas.com/content/sastraining/author/robertallison/

- The Graphics Samples Output Gallery is a collection of graphs organized by SAS procedure. The graphs link to the source code in SAS Samples & Notes. The gallery is maintained by SAS Technical Support.

  http://support.sas.com/sassamples/graphgallery/index.html

- Samples & SAS Notes contains an abundance of searchable examples. You can browse by topic, search for a particular note or a particular technology such as the name of a procedure, and conduct other searches.

  http://support.sas.com/notes/index.html

In addition, you can share your questions, suggestions, and experiences related to graphics on the SAS/GRAPH and ODS Graphics community site. See https://communities.sas.com/t5/SAS-GRAPH-and-ODS-Graphics/bd-p/sas_graph.

Chapter 2

Using Run-Group Processing

Using Run-Group Processing

You can use RUN-group processing with the GMAP procedure to produce multiple maps without restarting the procedure every time.

To use RUN-group processing, you start the procedure and then submit multiple RUN-groups. A *RUN-group* is a group of statements that contains at least one action statement and ends with a RUN statement. The procedure can contain other SAS statements such as BY, GOPTIONS, LEGEND, TITLE, or WHERE. As long as you do not terminate the procedure, it remains active and you do not need to resubmit the PROC statement.

To end RUN-group processing and terminate the procedure, submit a QUIT statement or start a new procedure. If you do not submit a QUIT statement, SAS/GRAPH does not terminate RUN-group processing until it reaches another step boundary.

Note: When using SAS/GRAPH with the ODS statement, it is best to use a QUIT statement after each procedure that uses RUN-group processing. Do this rather than relying on a new procedure to end the processing. Running too many procedures without an intervening QUIT statement can use up too much memory. Also, note that failing to submit a QUIT statement before submitting an ODS CLOSE statement results in the process memory not being freed at all.

RUN-group Processing with Global and Local Statements

Global statements and NOTE statements that are submitted in a RUN-group affect all subsequent RUN-groups until you cancel the statements or exit the procedure.

RUN-group Processing with BY Statements

BY statements persist in exactly the same way as global and local statements. Suppose you submit a BY statement within a RUN-group. Then the BY-group processing produces a separate graph for each value of the BY variable for the RUN-group in which you submit it. This continues for all subsequent RUN-groups until you cancel the BY statement or exit the procedure. Thus, as you submit subsequent action statements, you continue to get multiple graphs (one for each value of the BY variable). For more information, see the "BY Statement" on page 19.

For an example that produces three choropleth maps for each value of the BY variables STATECODE and STATENAME within a RUN-group, see "Example 11: Using Traditional Map Data to Produce a Drilldown Choropleth Map" on page 305.

RUN-group Processing with the WHERE Statement

The WHERE statement enables you to graph only a subset of the data in the input data set. If you submit a WHERE statement with a RUN-group, the WHERE definition remains in effect for all subsequent RUN-groups until you exit the procedure or reset the WHERE definition.

Using a WHERE statement with RUN-group processing follows most of the same rules as using the WHERE statement outside of RUN-group processing with one exception. With a procedure that is using an Annotate data set, the following requirements must be met:

- The ANNOTATE= option must be included in the action statement.
- The WHERE variable must occur in both the input data set and the Annotate data set.

Part 2

SAS/GRAPH Mapping Statements and System Options

Chapter 3

SAS/GRAPH Mapping Statements

Overview of Global Statements

SAS/GRAPH programs can use some of the SAS language statements that you typically use with the Base SAS procedures or with the DATA step, such as LABEL, WHERE, and FORMAT. These statements are described in the *SAS Statements: Reference*.

In addition, SAS/GRAPH has its own set of statements that affect only graphics output generated by the SAS/GRAPH procedures and the graphics Annotate facility. Most of these statements are *global statements*. That is, they can be specified anywhere in your program and remain in effect until explicitly changed, overridden, or canceled. These are the SAS/GRAPH global statements:

FOOTNOTE
adds footnotes to graphics output. This statement is like the TITLE statement and is described in that section.

GOPTIONS
submits graphics options that control the appearance of graphics elements by specifying characteristics such as colors, fill patterns, fonts, or text height. Graphics options can also temporarily change device settings.

LEGEND
modifies the appearance and position of legends generated by procedures that produce charts, plots, and maps.

NOTE
adds text to the graphics output. This statement is an exception because it is not global but local, meaning that it must be submitted within a procedure. Otherwise, the NOTE statement is like the TITLE statement and is described in that section.

PATTERN
controls the color and fill of patterns assigned to areas in charts, maps, and plots.

TITLE
adds titles to graphics output. The section describing the TITLE statement includes the FOOTNOTE and NOTE statements.

The above statements are described in this chapter, as well as the following two Base language statements that have a special effect when used with SAS/GRAPH procedures:

BY
processes data according to the values of a classification (BY) variable and produces a separate graph for each BY-group value. This statement is not a global statement. It must be specified within a DATA step or a PROC step.

ODS HTML
generates one or more files written in Hypertext Markup Language (HTML). If you use it with SAS/GRAPH procedures, you can specify one of the device drivers GIF, ACTIVEX, or JAVA. ACTIVEX and JAVA are available only with GMAP. With the GIF device driver, the graphics output is stored in GIF files. With the ACTIVEX device driver, graphics output is stored as XML input to ActiveX controls. With the JAVA device driver, graphics output is stored as XML input to Java applets. The HTML files that are generated reference the graphics output. When viewed with a web browser, the HTML files can display graphics and non-graphics output together on the same web page.

For more information about the BY, LABEL, OPTIONS, and WHERE statements in Base SAS software, see *SAS Statements: Reference*.

Specifying Units of Measurement

When the syntax of an option includes *units*, use one of these:

CELLS
character cells

CM
centimeters

IN
inches

PCT
percentage of the graphics output area

PT
points

Note: Java applets does not support CM, IN, or PT.

If you omit *units*, a unit specification is derived in this order:

1. The GUNIT= option in a GOPTIONS statement.
2. The default unit, CELLS.

Dictionary

BY Statement

Processes data and orders output according to the BY group.

Used by: GMAP, GREDUCE

Type: DATA step statement

Syntax

BY <DESCENDING> *variable-1* <<DESCENDING> *variable-2* …> <NOTSORTED>;

Required Argument

variable
specifies the variable that the procedure uses to form BY groups. You can specify more than one variable. By default, the procedure expects observations in the data set to be sorted in ascending order by all the variables that you specify or to be indexed appropriately.

Optional Arguments

DESCENDING
indicates that the data set is sorted in descending order by the specified variable. The option affects only the variable that immediately follows the option name, and must be repeated before every variable that is not sorted in ascending order. For example, this BY statement indicates that observations in the input data set are arranged in descending order of VAR1 values and ascending order of VAR2 values:

```
by descending var1 var2;
```

This BY statement indicates that the input data set is sorted in descending order of both VAR1 and VAR2 values:

```
by descending var1 descending var2;
```

NOTSORTED
specifies that observations with the same BY value are grouped together, but are not necessarily sorted in alphabetical or numeric order. The observations can be grouped in another way (for example, in chronological order).

NOTSORTED can appear anywhere in the BY statement and affects all variables specified in the statement. NOTSORTED overrides DESCENDING if both appear in the same BY statement.

The requirement for ordering or indexing observations according to the values of BY variables is suspended when you use the NOTSORTED option. In fact, the procedure does not use an index if you specify NOTSORTED. For NOTSORTED, the procedure defines a BY group as a set of contiguous observations that have the same values for all BY variables. Observations with the same value for the BY variables might not be contiguous. The procedure treats each new value that it encounters as

the first observation in a new BY group. The procedure creates a graph for that value, even if it is only one observation.

Details

Description: BY Statement

The BY statement divides the observations from an input data set into groups for processing. Each set of contiguous observations with the same value for a specified variable is called a *BY group*. A variable that defines BY groups is called a *BY variable* and is the variable that is specified in the BY statement. When you use a BY statement, the graphics procedure performs the following operations:

- processes each group of observations independently
- generates a separate graph or output for each BY group
- automatically adds a heading called a *BY line* to each graph identifying the BY group represented in the graph
- adds BY statement information below the **Description** field of the catalog entry

By default, the procedure expects the observations in the input data set to be sorted in ascending order of the BY variable values.

Note: The BY statement in SAS/GRAPH is essentially the same as the BY statement in Base SAS. However, the effect on the output is different when it is used with SAS/GRAPH procedures.

Note: In the GREMOVE procedure, the BY variables in the input map data set become the ID variables for the output map data set.

Preparing Data for BY-Group Processing

Unless you specify the NOTSORTED option, observations in the input data set must be in ascending numeric or alphabetic order. To prepare the data set, you can sort it with the SORT procedure using the same BY statement that you plan to use in the target SAS/GRAPH procedure. Or you can create an appropriate index on the BY variables.

If the procedure encounters an observation that is out of the proper order, it issues an error message.

If you need to group data in some other order, you can still use BY-group processing. To do so, process the data so that observations are arranged in contiguous groups that have the same BY-variable values and specify the NOTSORTED option in the BY statement.

For an example of sorting the input data set, see "Using BY-group Processing to Generate a Series of Charts" in *SAS/GRAPH: Reference*.

Controlling BY Lines

Understanding Default Behavior

By default, the BY statement prints a BY line above each graph that contains the variable name followed by an equal sign and the variable value. For example, if you specify BY SITE in the procedure, the default heading when the value of SITE is `London` would be SITE=London.

Suppressing the BY Line

To suppress the entire BY line, use the NOBYLINE option in an OPTION statement or specify HBY=0 in the GOPTIONS statement. See "Using BY-group Processing to Generate a Series of Charts" in *SAS/GRAPH: Reference*.

Suppressing the Name of the BY Variable

You can suppress the variable name and the equal sign in the heading and leave only the BY value. Use the LABEL statement to assign a null label ("00"X) to the BY variable. For example, this statement assigns a null label to the SITE variable:

```
label site="00"x;
```

Controlling the Appearance of the BY Line

To control the color, font, and height of the BY lines, use the following graphics options in a GOPTIONS statement:

CBY=*BY-line-color*
specifies the color for BY lines.

FBY=*font*
specifies the font for BY lines.

HBY=*n<units>*
specifies the height for BY lines.

For a description of each option, see the "Graphics Options and Device Parameters Dictionary " in *SAS/GRAPH: Reference*.

Naming the Catalog Entries

The catalog entries generated with BY-group processing always use incremental naming. This means that the first entry created by the procedure uses the base name and subsequent entries increment that name. The base name is either the default entry name for the procedure (for example, GMAP) or the name specified with the NAME= option in the action statement. Incrementing the base name automatically appends a number to each subsequent entry (for example, GMAP1, GMAP2, and so on). See "Specifying the Catalog Name and Entry Name for Your GRSEGs " in *SAS/GRAPH: Reference*. For an example of incremented catalog names, see "Combining Graphs and Reports in a Web Page" in *SAS/GRAPH: Reference*.

Using the BY Statement

Overview

This section describes the following:

- the effect of BY-group processing on the GMAP procedure
- the interaction between BY-group and RUN-group processing
- the requirements for using BY-group processing with the Annotate facility
- how to include BY information in legend labels and values, and titles, notes, and footnotes
- how patterns are assigned to BY-groups
- the effect of using BY-group processing with the ODS HTML statement

For additional information about any of these topics, refer to the appropriate chapter.

Using the BY Statement with the GMAP Procedure

By default, BY-group processing affects both the map data set and the response data set. This means that you get separate, individual output for each map area common to both data sets. For example, suppose the map data set REGION contains six states and the response data set contains the same six states. If you specify BY STATE in the GMAP procedure, the resulting output is six graphs with one state on each graph.

If you use the ALL option in the PROC GMAP statement and you also use the BY statement, you get one output for each map area in the response data set. However, that output displays all the map areas in the map data set. Only one map area per output contains response data information; the others are empty. For example, suppose you create a block map using the data sets REGION and SALES, specify BY STATE, and include the ALL option in the PROC GMAP statement. The resulting output is six graphs with six states on each graph. One state per graph has a block; the remaining five are empty. The UNIFORM option applies colors and heights uniformly across all BY-groups.

Using the BY Statement with the RUN Groups

If you use the BY statement with a GMAP procedure that processes data and supports RUN-group processing, then each time you submit an action statement or a RUN statement, you get a separate map for each value of the BY variable. For an example that produces three choropleth maps for each value of the BY variables STATECODE and STATENAME within a RUN-group, see "Example 11: Using Traditional Map Data to Produce a Drilldown Choropleth Map" on page 305.

The BY statement stays in effect for every subsequent RUN group until you submit another BY statement or exit the procedure. Variables in subsequent BY statements replace any previous BY variables.

You can also turn off BY-group processing by submitting a null BY statement (BY;) in a RUN group. Do this with care however, because the null BY statement turns off BY-group processing and the RUN group generates a graph.

For more information, see Chapter 2, "Using Run-Group Processing," on page 13.

Using the BY Statement with the Annotate Facility

If a procedure that is using BY-group processing also specifies annotation with the ANNOTATE= option in the PROC statement, the same annotation is applied to every graph generated by the procedure.

If you specify annotation with the ANNOTATE= option in the action statements for a procedure, the BY-group processing is applied to the Annotate data set. In this way, you can customize the annotation for the output from each BY group. Include the BY variable in the Annotate data set and use each BY-variable value as a condition for the annotation to be applied to the output for that value.

Using the BY Statement with LEGEND, TITLE, FOOTNOTE, and NOTE Statements

LEGEND statements can automatically include the BY variable name or BY variable value in the text that they produce for labels, reference labels, values for major tick marks, and legend labels and values. In addition, TITLE, FOOTNOTE, and NOTE statements can automatically include the BY lines in the text that they produce. To insert BY line information into the text strings used by these statements, use the appropriate #BYVAR, #BYVAL, and #BYLINE substitution options.

Using the BY Statement with PATTERN Definitions

By default, when using a BY statement, the map for each BY group uses the same patterns in their defined order. For example, the BY variable contains four values and there are two response levels for each BY value. In this case the PATTERN1 and PATTERN2 statements are used for each map. Each BY-group starts over with PATTERN1. The UNIFORM option in the GMAP procedure changes this behavior.

FOOTNOTE Statement

Writes up to 10 lines of text at the bottom of the graph.

Type: Global

See: "TITLE, FOOTNOTE, and NOTE Statements" on page 56

Syntax

FOOTNOTE<1 …10> <*text-argument(s)*>;

GOPTIONS Statement

Temporarily sets default values for many graphics attributes and device parameters used by SAS/GRAPH procedures.

Used by: all statements and procedures in a SAS session

Type: Global

Syntax

GOPTIONS <*options-list*>;

Optional Argument

options-list
can be one or more options as listed and described in the "Graphics Options and Device Parameters Dictionary " in *SAS/GRAPH: Reference*.

Details

Description: GOPTIONS Statement

The GOPTIONS statement specifies values for *graphics options*. Graphics options control characteristics of the graph, such as size, colors, type fonts, fill patterns, and symbols. If GOPTIONS are specified, they override the default style. In addition, they affect the settings of device parameters, which are defined in the device entry. Device parameters control such characteristics as the appearance of the display, the type of output produced, and the destination of the output.

The GOPTIONS statement enables you to change these settings temporarily, either for a single graph or for the duration of your SAS session. You can use the GOPTIONS statement to do the following tasks:

- override default values for graphics options that control either graphics attributes or device parameters for a single graph or for an entire SAS session
- reset individual graphics options or all graphics options to their default values
- cancel definitions for FOOTNOTE, PATTERN, and TITLE statements

To change device parameters permanently, you must use the GDEVICE procedure to modify the appropriate device entry or to create a new one. See the "The GDEVICE Procedure" in *SAS/GRAPH: Reference* for details.

To review the current settings of all graphics options, use the GOPTIONS procedure. See the "The GOPTIONS Procedure" in *SAS/GRAPH: Reference* for details.

Using the GOPTIONS Statement

GOPTIONS statements are global and can be located anywhere in your SAS program. However, for the graphics options to affect the output from a procedure, the GOPTIONS statement must execute before the procedure.

With the exception of the RESET= option, graphics options can be listed in any order in a GOPTIONS statement. The RESET= option should be the first option in the GOPTIONS statement.

A graphics option remains in effect until you specify the option in another GOPTIONS statement, use the RESET= option to reset the values, or end the SAS session. When a session ends, the values of the graphics options return to their default values.

Graphics options are additive. That is, the value of a graphics option remains in effect until the graphics option is explicitly changed or reset or until your SAS session ends. Graphics options remain in effect even after you submit additional GOPTIONS statements specifying different options.

To reset an individual option to its default value, submit the option without a value (a null graphics option.) You can use a comma (but it is not required) to separate a null graphics option from the next one. For example, this GOPTIONS statement sets the values for the background color, the text height, and the text font:

```
goptions cback=blue htext=6 pct ftext="Albany AMT";
```

Note: When there is a space in the font name, surround the name in quotation marks. The quotation marks also indicate use of a system font rather than a software graph font.

To reset only the background color specification to the default and keep the remaining values, use this GOPTIONS statement:

```
goptions cback=;
```

To reset all graphic options to their default values, specify RESET=GOPTIONS:

```
goptions reset=goptions;
```

Alternatively, you can use RESET=ALL, but it also cancels any global statement definitions in addition to resetting all graphics options to default values.

Graphics Option Processing

You can control many graphics attributes through statement options, graphics options, device parameters, or a combination of these. SAS/GRAPH searches these places to determine the value to use, stopping at the first place that gives it an explicit value:

1. statement options
2. the value of the corresponding graphics option
3. the value of a device parameter found in the catalog entry for your device driver

Note: Not every graphics attribute can be set in all three places. See the statement and procedure chapters for the options that can be used with each.

Some graphics options are supported for specific devices or operating environments only. See the SAS Help facility for SAS/GRAPH or the SAS companion for your operating environment for more information.

LEGEND Statement

Controls the location and appearance of legends on maps.

Used by: GMAP

Type: Global

Syntax

LEGEND<1 …99> <*option(s)*>;

Summary of Optional Arguments

Appearance options

ACROSS=*number-of-columns*
: specifies the number of columns to use for legend entries.

CBLOCK=*block-color*
: generates and colors a three-dimensional block effect behind the legend.

CBORDER=*frame-color*
: draws a colored frame around the legend.

CFRAME=*background-color*
: specifies the background color of the legend.

CSHADOW=*shadow-color*
: generates and colors a drop shadow behind the legend.

DOWN=*number-of-rows*
: specifies the number of rows to use for legend entries.

FRAME | NOFRAME
: draws a frame around the legend.

FWIDTH=*thickness-factor*
: specifies the thickness of the frame, where *thickness-factor* is a number.

REPEAT=1 | 2 | 3
: specifies how many times the plot symbol is repeated in the legend.

ROWMAJOR | COLMAJOR
: specifies the arrangement of legend entries when there are multiple rows and multiple columns.

SHAPE=BAR(*width*<*units*>,*height*<*units*>) <*units*>
: specifies the size and shape of the legend values displayed in each legend entry.

SPACE=*value-spacing*
: specifies the amount of space between individual legend values along a horizontal baseline.

Position options

MODE=PROTECT | RESERVE | SHARE
: specifies whether the legend is drawn in the procedure output area or whether legend elements can overlay other graphics elements.

OFFSET=(<*x*><,*y*>)<*units*> | (<*x* <*units*>><,*y* <*units*>>)
specifies the distance to move the entire legend.

ORIGIN=(<*x*><,*y*>)<*units*> | (<*x* <*units* >><,*y* <*units*>>)
specifies the *x* and *y* coordinates of the lower left corner of the legend box.

POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT> <OUTSIDE | INSIDE>)
positions the legend on the graph.

Text options

LABEL=(*text-argument(s)*) | NONE
modifies a legend label.

ORDER=(*value-list*) | DESCENDING
selects or orders the legend values that appear in the legend.

SPLIT="*split-char(s)*"
specifies one or more characters that the LEGEND statement uses to break a text description string into multiple lines.

VALUE=(*text-argument(s)*) | NONE
modifies the legend value descriptions.

Optional Arguments

ACROSS=*number-of-columns*
specifies the number of columns to use for legend entries. If there are multiple rows and columns in a legend, use the ROWMAJOR and COLMAJOR options to specify the arrangement of legend entries. Specify the ROWMAJOR option to arrange entries (from lowest to highest) starting from left to right, and then top to bottom. Specify the COLMAJOR option to arrange entries starting from top to bottom, and then left to right.

See " ROWMAJOR | COLMAJOR" on page 32

Example "Example 2: Creating a Simple Web Page with the ODS HTML Statement" on page 77

CBLOCK=*block-color*
generates and colors a three-dimensional block effect behind the legend. The size and position of the block are controlled by the graphics option OFFSHADOW=(*x*,*y*)..

The Java applet treats the CBLOCK option like the CSHADOW option.

Restrictions Not supported by Java.

The CBLOCK= and CSHADOW= options are mutually exclusive. If both are present, SAS/GRAPH software uses the last one specified.

Interaction The CBLOCK= option is usually used in conjunction with the FRAME, CFRAME=, or CBORDER= options.

See "OFFSHADOW" in *SAS/GRAPH: Reference*

"Creating Drop Shadows and Block Effects" on page 42

CBORDER=*frame-color*
draws a colored frame around the legend.

Style reference Color attribute of the GraphBorderLines graph element

Restriction This option overrides the FRAME option.

Interaction CBORDER= can be used in conjunction with the CFRAME= option.

CFRAME=*background-color*
specifies the background color of the legend. This option overrides the FRAME option.

Style reference Color attribute of the GraphLegendBackground graph element

Restriction If both the CFRAME= and FRAME= options are specified, only the solid background produced by the CFRAME= option is displayed.

Interaction The CFRAME= option can be used in conjunction with the CBORDER= option.

CSHADOW=*shadow-color*
generates and colors a drop shadow behind the legend. The size and position of the shadow is controlled by the graphics option OFFSHADOW=(*x,y*).

Restriction The CSHADOW= and CBLOCK= options are mutually exclusive. If both are present, SAS/GRAPH uses the last one specified.

Interaction The CSHADOW= option is usually specified in conjunction with the FRAME, CFRAME=, or CBORDER= options.

See "OFFSHADOW" in *SAS/GRAPH: Reference*

"Creating Drop Shadows and Block Effects" on page 42

DOWN=*number-of-rows*
specifies the number of rows to use for legend entries. If there are multiple rows and columns in a legend, use the ROWMAJOR and COLMAJOR options to specify the arrangement of legend entries. Specify the ROWMAJOR option to arrange entries (from lowest to highest) starting from left to right, and then top to bottom. Specify the COLMAJOR option to arrange entries starting from top to bottom, and then left to right.

Default When there are multiple rows and columns in a legend, the ROWMAJOR option is the default

See " ROWMAJOR | COLMAJOR" on page 32

FRAME | NOFRAME
draws a frame around the legend. The color of the frame is the first color in the color list. NOFRAME suppresses the drawing of a frame, and is the default.

FWIDTH=*thickness-factor*
specifies the thickness of the frame, where *thickness-factor* is a number. The thickness of the line increases directly with *thickness-factor*. By default, FWIDTH=1.

Restriction Not supported by Java and ActiveX

LABEL=(*text-argument(s)*) | NONE

modifies a legend label. *Text-argument(s)* defines the appearance or the text of a legend label, or both. NONE suppresses the legend label in most instances. By default, the text of the legend label is either the variable name or a previously assigned variable label (except in the case of GPLOT with OVERLAY. In that case, the default label is "PLOT"). You can use an overlay variable and suppress the legend label that would display that variable name. Specify a SAS software font to generate the unprintable hexadecimal character of '00'x, as shown in this example:

```
legend1 label=(font=swiss '00'x);
```

Text-argument(s) can be one or more of these:

"text-string"

provides up to 256 characters of label text. Enclose each string in quotation marks. Separate multiple strings with blanks.

In addition, if you have a BY statement and you specify the variable that it names, you can embed one or both of the following in the string:

#BYVAL*n* | #BYVAL(*BY-variable-name*)

substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value produced by the statement. Specify the variable with one of these:

n

specifies which variable in the BY statement #BYVAL should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAL2 specifies the second variable in the BY statement.

BY-variable-name

names the BY variable. For example, #BYVAL(YEAR) specifies the BY variable, YEAR. *Variable-name* is not case sensitive.

Examples

"Combining Graphs and Reports in a Web Page" in *SAS/GRAPH: Reference*

#BYVAR*n* | #BYVAR(*BY-variable-name*)

substitutes the name of the BY variable or label associated with the variable (whatever the BY line would normally display) for #BYVAR in the text string and displays the name or label produced by the statement. Specify the variable with one of these:

n

specifies which variable in the BY statement #BYVAR should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAR2 specifies the second variable in the BY statement.

BY-variable-name

names the BY variable. For example, #BYVAR(SITES) specifies the BY variable, SITES. *Variable-name* is not case sensitive.

Note A BY variable name displayed is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

See "Substituting BY Line Values in a Text String" on page 42

text-description-suboption

modifies a characteristic such as the font, color, or size of the text strings that follows it. *Text-description-suboption* can be as follows:

- ANGLE=*degrees*
- COLOR=*text-color*
- FONT=*font* | NONE
- HEIGHT=*text-height* <*units*>
- JUSTIFY=LEFT | CENTER | RIGHT
- POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT>)
- ROTATE=*degrees*

For a complete description of these suboptions, see "Text Description Suboptions" on page 35.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

Style reference

Color attribute of the GraphLabelText graph element

Restrictions

Partially supported by Java and ActiveX

#BYVAL or #BYVAR substitution in a text string is not available in the Annotate facility. The reason is that BY lines are not created in a DATA step.

When there is not enough room in the output to display a long label text, the legend entry and label can be suppressed. In that event warning messages are issued in the SAS log.

Note

The Java applet does not support the POSITION= suboption—it draws legend labels at the top left of the legend. Also, it does not support multiple values for the JUSTIFY= suboption (only the first is honored). The ActiveX control supports the POSITION= option but does not support multiple values for the JUSTIFY suboption (only the first is honored).

Examples

"Example 2: Creating a Simple Web Page with the ODS HTML Statement" on page 77

MODE=PROTECT | RESERVE | SHARE

specifies whether the legend is drawn in the procedure output area or whether legend elements can overlay other graphics elements. If the space required to display the legend exceeds that of the space required to display the graph, a warning is issued and the legend is suppressed. The MODE= option can take one of these values:

PROTECT

draws the legend in the procedure output area, but a *blanking area* surrounds the legend, preventing other graphics elements from being displayed in the legend. (A blanking area is a protected area in which no other graphics elements are displayed.)

RESERVE
takes space for the legend from the procedure output area, thereby reducing the amount of space available for the graph. If MODE=RESERVE is specified in conjunction with OFFSET=, the legend can push the graph off the graphics output area. RESERVE is valid only when POSITION=OUTSIDE. If POSITION=INSIDE is specified, a warning is issued and MODE= value is changed to PROTECT.

SHARE
draws the legend in the procedure output area. If the legend is positioned over elements of the graph itself, both graphics elements and legend elements are displayed.

By default, MODE=RESERVE unless POSITION=INSIDE. In this case, the default changes to MODE=PROTECT.

If the MODE=PROTECT option is used with labels on the midpoint axis, the axis labels might overlap the legend. In that case, if a device-resident font is used as the label font, the device-resident font for each label that overlaps the legend is replaced with a SAS/GRAPH font. This includes the axis label and the tick mark labels. Because device-resident fonts do not support clipping, a SAS/GRAPH font must be substituted in that case. To correct an overlap condition, you can use the LABEL=NONE and VALUE=NONE options on your midpoint axis statement to suppress the axis labels. Another choice is to use the positioning options on your LEGEND statement to reposition the legend.

Restriction Not supported by Java and ActiveX

See "Positioning the Legend" on page 40

OFFSET=(<*x*><,*y*>)<*units*> | (<*x* <*units*>><,*y* <*units*>>)
specifies the distance to move the entire legend. *x* is the number of units to move the legend right (positive numbers) or left (negative numbers). *y* is the number of units to move the legend up (positive numbers) or down (negative numbers).

To set only the *x* offset, specify one value, with or without a following comma:

```
offset=(4 cm,)
```

To set both the *x* and *y* offset, specify two values, with or without a comma separating them:

```
offset=(2 pct, 4 pct)
```

To set only the *y* offset, specify one value preceded by a comma:

```
offset=(,-3 pct)
```

The OFFSET= option is usually used in conjunction with the POSITION= option to adjust the position of the legend. Moves are relative to the location specified by the POSITION= option, with OFFSET=(0,0) representing the initial position. You can also apply the OFFSET= option to the default legend position.

The OFFSET= option is unnecessary with the ORIGIN= option because the ORIGIN= option explicitly positions the legend and requires no further adjustment. However, if you specify both options, the OFFSET= values are added to the ORIGIN= values, and the LEGEND is positioned accordingly.

Restriction Not supported by Java and ActiveX

See "Positioning the Legend" on page 40

" POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT> <OUTSIDE | INSIDE>)" on page 32

ORDER=(*value-list*) | DESCENDING

selects or orders the legend values that appear in the legend. DESCENDING specifies that the legend values appear in the legend in reverse of the default order, sorted from maximum to minimum value. Alternatively, use *value-list* to select only those legend values that you want to display and by their placement in the list, order where they appear in the legend. The way you specify *value-list* depends on the type of variable that generates the legend:

- For numeric variables, *value-list* is either an explicit list of values, or a starting value 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*>>

  If a numeric variable has an associated format, the specified values must be the unformatted values.
- For character variables, *value-list* is a list of unique character values enclosed in quotation marks and separated by blanks:
  - *"value-1"<" value-2" ..." value-n">*

  If a character variable has an associated format, the specified values must be the formatted values.

For a complete description of *value-list*, see the ORDER= option in the "AXIS Statement" in *SAS/GRAPH: Reference*.

Even though the ORDER= option controls whether a legend value is displayed and where it appears, the VALUE= option controls the text that the legend value displays.

Restrictions Not supported by Java and ActiveX

Up to 256 characters in a variable value might be displayed.

ORIGIN=(<*x*><,*y*>)<*units*> | (<*x* <*units* >><,*y* <*units*>>)

specifies the *x* and *y* coordinates of the lower left corner of the legend box. The ORIGIN= option explicitly positions the legend anywhere on the graphics output area. It is possible to run a legend off the page or overlay the graph.

To set only the *x* coordinate, specify one value, with or without a following comma:

```
origin=(4 cm,)
```

To set both the *x* and *y* coordinates, specify two values, with or without a comma separating them:

```
origin=(2 pct, 4 pct)
```

To set only the *y* coordinate, specify one value preceded by a comma:

```
origin=(,3 pct)
```

The ORIGIN= option overrides the POSITION= option if both are used. Although using the OFFSET= option with the ORIGIN= option is unnecessary, if the OFFSET= option is also specified, it is applied after the ORIGIN= request has been processed.

Restriction Not supported by Java and ActiveX

See "Positioning the Legend" on page 40

POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT> <OUTSIDE | INSIDE>)

positions the legend on the graph. Values for POSITION= are as follows:

OUTSIDE or INSIDE
specifies the location of the legend in relation to the axis area.

BOTTOM or MIDDLE or TOP
specifies the vertical position.

LEFT or CENTER or RIGHT
specifies the horizontal position.

By default, POSITION=(BOTTOM CENTER OUTSIDE). You can change one or more settings. If you supply only one value, then the parentheses are not required. If you specify two or three values and omit the parentheses, SAS/GRAPH accepts the first value and ignores the others.

Once you assign the initial legend position, you can adjust it with the OFFSET= option.

The ORIGIN= option overrides the POSITION= option. The value of the MODE= option can affect the behavior of the POSITION= option.

Restriction Partially supported by Java

Note The Java applet defaults to BOTTOM-CENTER and supports all possible combinations of BOTTOM | MIDDLE | TOP with LEFT | CENTER | RIGHT except for MIDDLE-CENTER (which would overwrite the map.) The Java applet does not support INSIDE.

See " OFFSET=(<*x*><,*y*>)<*units*> | (<*x* <*units*>><,*y* <*units*>>)" on page 30

" MODE=PROTECT | RESERVE | SHARE" on page 29

REPEAT=1 | 2 | 3

specifies how many times the plot symbol is repeated in the legend. Valid values are 1 to 3. The default value is 3 .

Restriction Not supported by Java or ActiveX

ROWMAJOR | COLMAJOR

specifies the arrangement of legend entries when there are multiple rows and multiple columns. Specify the ROWMAJOR option (the default) to arrange entries (from lowest to highest) starting from left to right, and then top to bottom. Specify the COLMAJOR option to arrange the entries starting from top to bottom, and then left to right.

See " ACROSS=*number-of-columns*" on page 26

" DOWN=*number-of-rows*" on page 27

SHAPE=BAR(*width*<*units*>,*height*<*units*>) <*units*>

specifies the size and shape of the legend values displayed in each legend entry. The SHAPE= value that you specify depends on which procedure generates the legend.

BAR(*width*,*height*)<*units*> | (*width*<*units*>,*height*<*units*>) <*units*>
is used with the GMAP procedure. Each legend value is a bar of the specified width and height.

Default *width* is 5, *height* is 0.8, and *units* are CELLS.

Note You can specify the width and height in units of character cells (CELLS), centimeters (CM), inches (IN), percentage of the graphics output area (PCT), or points (PT). There are approximately 72 points in an inch.

Tip You can specify *units* for the *width*,*height* pair or for either or both of the individual coordinates.

SPACE=*value-spacing*
specifies the amount of space between individual legend values along a horizontal baseline. *Value-spacing* can be any nonnegative number, including decimal values. Units are only character cells. You cannot specify a unit of measure, such as inches or percent. The LEGEND statement calculates spacing based on the available display area, and the number and width of legend entries. If the spacing requested exceeds the display area, the legend values and their descriptions are rearranged to fit.

The SPACE= option is ignored if the following is true:

- SPACE=0 is specified.
- The specified spacing is requested in conjunction with vertically stacked columns.

Restriction Not supported by Java or ActiveX

SPLIT="*split-char(s)*"
specifies one or more characters that the LEGEND statement uses to break a text description string into multiple lines. *Split-char(s)* can be any character value that can be specified in a SAS character variable. Do not delimit when specifying multiple split characters. When the LEGEND statement encounters the split character, it automatically breaks the value at that point and continues on the next line. For example, suppose the legend text description contains the string **`Berlin, Germany/Europe`**, and you specify SPLIT=",/". The legend breaks the text string into top-aligned lines as follows:

```
Berlin
Germany
Europe
```

If the LEGEND statement does not encounter a specified split character, no break in the text occurs, and no warning or error is issued.

Restrictions Not supported by Java and ActiveX

Not supported by the GRADAR procedure

Up to 31 split characters are supported. If you specify more than 31 characters, only the first 31 are honored.

Note The split characters themselves are not displayed.

Example "Creating a Drill-Down HTML Presentation for the Web" in *SAS/GRAPH: Reference*

VALUE=(*text-argument(s)*) | NONE

modifies the legend value descriptions. *Text-argument(s)* defines the appearance or the text of the value descriptions. By default, value descriptions are the values of the variable that generates the legend or an associated format value. Numeric values are right-justified and character values are left-justified.

NONE suppresses the value descriptions although the legend values (bars, lines, and so on) are still displayed. *Text-argument(s)* can be one or more of these:

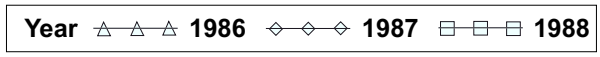
"text-string"

provides up to 256 characters of text for the value description. Enclose each string in quotation marks. Separate multiple strings with blanks. Value description text that is too long to fit in a legend area can result in the entire value not being displayed.

Specified text strings are assigned to the legend values in order. If you submit only one string, only the first legend entry uses the value of that string. If you specify multiple strings, the first string is the text for the first entry; the second string is the text for the second entry; and so on. For example, this specification produces legend entries like those shown in Figure 3.1 on page 34:

```
value=("1986" "1987"
"1988")
```

Figure 3.1 *Specifying Value Descriptions with the VALUE= Option*



In addition, if you have a BY statement and you specify the variable that it names, you can embed one or both of the following in the string:

#BYVAL*n* | #BYVAL(*BY-variable-name*)

substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value produced by the statement. Specify the variable with one of these:

n

specifies which variable in the BY statement #BYVAL should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAL2 specifies the second variable in the BY statement.

BY-variable-name

names the BY variable. For example, #BYVAL(YEAR) specifies the BY variable, YEAR. *Variable-name* is not case sensitive.

Examples

"Combining Graphs and Reports in a Web Page" in *SAS/GRAPH: Reference*

#BYVAR*n* | #BYVAR(*BY-variable-name*)

substitutes the name of the BY variable or label associated with the variable (whatever the BY line would normally display) for #BYVAR in the text string and displays the name or label produced by the statement. Specify the variable with one of these:

n

specifies which variable in the BY statement #BYVAR should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAR2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAR(SITES) specifies the BY variable, SITES. *Variable-name* is not case sensitive.

Note A BY variable name displayed is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

See "Substituting BY Line Values in a Text String" on page 42

text-description-suboption
modifies a characteristic such as the font, color, or size of the text string(s) that follows it. *Text-description-suboption* can be as follows:

- ANGLE=*degrees*
- COLOR=*text-color*
- FONT=*font* | NONE
- HEIGHT=*text-height* <*units* >
- JUSTIFY=LEFT | CENTER | RIGHT
- POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT>)
- ROTATE=*degrees*
- TICK=*n*

Place text description suboptions before the text strings that they modify. Suboptions not followed by a text string affect the default values. To specify and describe the text for individual values or to produce multi-line text, use the TICK= suboption.

Specify as many text strings and text description suboptions as you want, but enclose them all in one set of parentheses.

To order or select legend entries, use the ORDER= option.

Restrictions Partially supported by Java and ActiveX

ActiveX control does not support changing the font in the middle of specifying descriptive text for a legend.

NONE is not supported by Java or ActiveX

#BYVAL or #BYVAR substitution in a text string is not available in the Annotate facility. The reason is that BY lines are not created in a DATA step.

See "Text Description Suboptions" on page 35

" ORDER=(*value-list*) | DESCENDING" on page 31

Text Description Suboptions

ANGLE=*degrees*
specifies the angle of the legend label or legend value description text with respect to the horizontal. A positive value for degrees moves the text counterclockwise; a negative value moves it clockwise. By default, ANGLE=0 (horizontal).

Note: Some settings of ANGLE= in the LEGEND statement might result in undesirable text positioning.

Alias A=

Restriction Not supported by Java and ACTIVEX

See "ROTATE=*degrees*" on page 38

COLOR=*text-color*
specifies the color of the text. If you omit the COLOR= suboption, a color specification is searched for in this order:

1. the CTEXT= option for the procedure
2. the CTEXT= option in a GOPTIONS statement
3. the color of the default style

Alias C=

FONT=*font* | NONE
specifies the font for the text. See "Specifying Fonts in SAS/GRAPH Programs" in *SAS/GRAPH: Reference* for information about specifying fonts. If you omit the FONT= suboption, a font specification is searched for in this order:

1. the FTEXT= option in a GOPTIONS statement
2. the default style font, NONE

Alias F=

Restriction ActiveX control does not support changing the font in the middle of specifying descriptive text for a legend.

HEIGHT=*text-height <units>*
specifies the height of the text characters in the number of units. By default, HEIGHT=1 CELL. If you omit the HEIGHT= suboption, a text height specification is searched for in this order:

1. the HTEXT= option in a GOPTIONS statement
2. the height specified by the default style

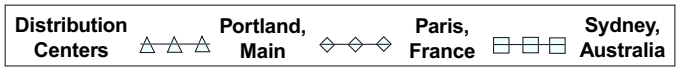
Alias H=

JUSTIFY=LEFT | CENTER | RIGHT
specifies the alignment of the text. The default for character variables is JUSTIFY=LEFT. The default for numeric variables is JUSTIFY=RIGHT. Associating a character format with a numeric variable does not change the default justification of the variable.

You can use the JUSTIFY= suboption to print multiple lines of text by repeating the suboption before the text string for each line. For example, this statement produces a legend label and value descriptions like those shown in Figure 3.2 on page 37:

```
legend label=(justify=c "Distribution"
              justify=c "Centers")
value=(tick=1 justify=c "Portland,"
              justify=c "Maine"
       tick=2 justify=c "Paris,"
              justify=c "France"
       tick=3 justify=c "Sydney,"
              justify=c "Australia");
```

Figure 3.2 *Specifying Multiple Lines of Text with the JUSTIFY= Suboption*



Place text description suboptions before the text strings that they modify.

Alias J=L | C | R

See " TICK=*n*" on page 39

POSITION=(<BOTTOM | MIDDLE | TOP> <LEFT | CENTER | RIGHT>)
places the legend label in relation to the legend entries. The POSITION= suboption is used only with the LABEL= option. By default, POSITION=LEFT.

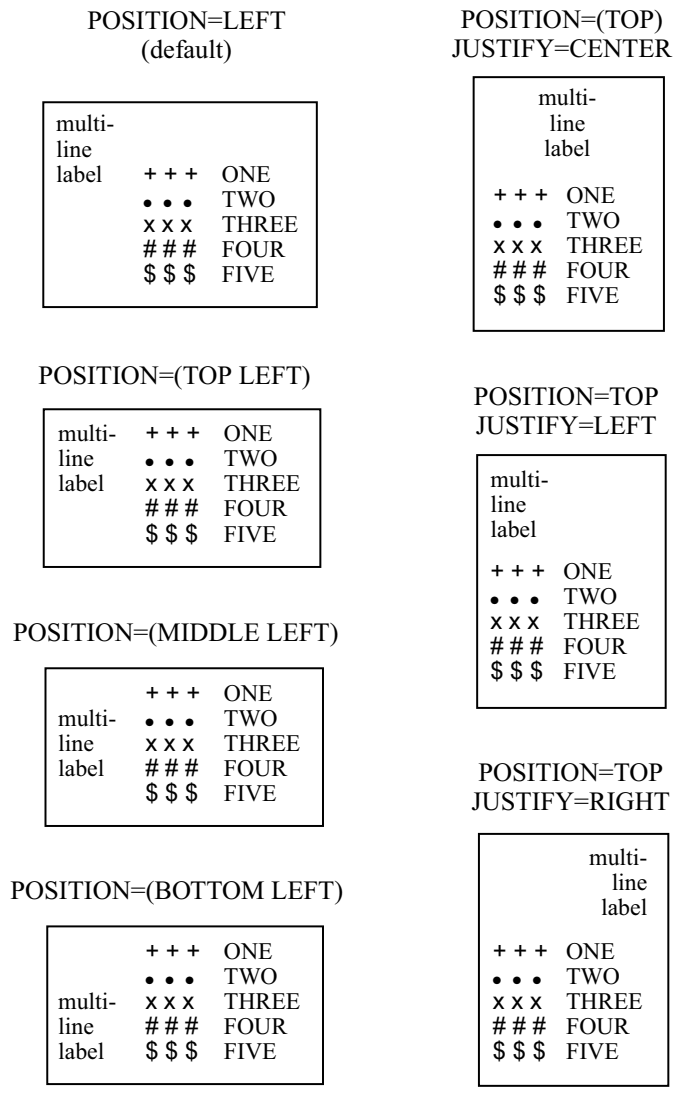
The parentheses are not required if only one value is supplied. If you specify two or three values and omit the parentheses, SAS/GRAPH accepts the first value and ignores the others.

Figure 3.3 on page 38 shows some of the ways the POSITION= suboption affects a multiple-line legend label in which the entries are stacked in a column (ACROSS=1). This figure uses a label specification such as the following:

```
label=("multi-"
        justify=left "line"
        justify=left "label"
        position=left)
```

In this specification, the POSITION= suboption specifies the default value, LEFT, which is represented by the first legend in the figure. The POSITION= value is indicated above each legend. The default justification is used unless you also use the JUSTIFY= suboption.

Figure 3.3 *Using the POSITION= Suboption with Multiple-line Legend Labels*



In addition, specifying POSITION=RIGHT mirrors the effect of POSITION=LEFT, and specifying POSITION=BOTTOM mirrors the effect of POSITION=TOP.

Restriction Not supported by Java. Partially supported by ActiveX

ROTATE=*degrees*

specifies the angle at which each character of text is rotated with respect to the *baseline* of the text string. A positive value for degree rotates the character counterclockwise; a negative value moves it clockwise. By default, ROTATE=0 (parallel to the *baseline*).

Note: Some settings of ROTATE= in the LEGEND statement might result in undesirable text positioning.

Alias R=

Restriction Not supported by Java and ACTIVEX

See "ANGLE=*degrees*" on page 35

TICK=*n*

specifies the nth legend entry. The TICK= suboption is used only with the VALUE= option to designate the legend entry whose text and appearance you want to modify. For example, to change the text of the third legend entry to **`Minneapolis`**, specify the following code:

```
value=(tick=3 "Minneapolis")
```

The characteristics of all other value descriptions remain unchanged.

If you use the TICK= suboption when you designate text for one legend entry, you must also use it when you designate text for any additional legend entries. For example, this option changes the text of both the second and third legend entries:

```
value=(tick=2 "Paris" tick=3 "Sydney")
```

If you omitted TICK=3, the text of the second legend entry would be **`ParisSydney`**.

Text description suboptions that precede the TICK= suboption affect all the value descriptions for the legend unless the same suboption (with a different value) follows a TICK= specification. Text description suboptions that follow the TICK= suboption affect only the specified legend entry. For example, suppose you specify this option for a legend with three entries:

```
value=(color=red font=swiss tick=2 color=blue)
```

The text of all three entries would use the Swiss font; the first and third entries would be red and only the second entry would be blue.

Alias T=

Details

Description: LEGEND Statement

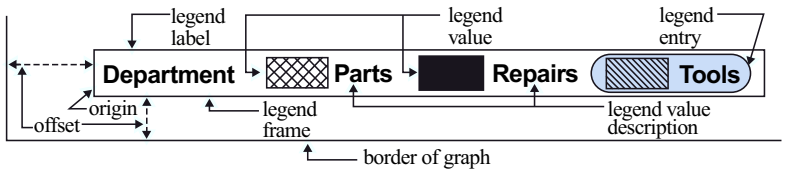
LEGEND statements specify the characteristics of a legend but do not create legends. The characteristics are as follows:

- the position and appearance of the legend box
- the text and appearance of the legend label
- the appearance of the legend entries, including the size and shape of the legend values
- the text of the labels for the legend values

LEGEND definitions are not automatically applied when a procedure generates a legend. Instead, they must be explicitly assigned with a LEGEND= option in the appropriate procedure statement.

The following figure illustrates the terms associated with the various parts of a legend.

Figure 3.4 *Parts of a Legend*



Using Text Description Suboptions

Text description suboptions affect all the strings that follow them unless the suboption is changed or turned off. If the value of a suboption is changed, the new value affects all the text strings that follow it. Consider this example:

```
label=(font=albany amt height=4 "Weight"
       justify=right height=3 "(in tons)")
```

FONT=ALBANY applies to both **Weight** and **(in tons)**. HEIGHT=4 affects **Weight**, but is respecified as HEIGHT=3 for **(in tons)**. JUSTIFY=RIGHT affects only **(in tons)**.

Using the LEGEND Statement

LEGEND statements can be located anywhere in your SAS program. They are global and remain in effect until canceled or until you end your SAS session. LEGEND statements are not applied automatically, and must be explicitly assigned by an option in the procedure that uses them.

You can define up to 99 different LEGEND statements. If you define two LEGEND statements of the same number, the most recently defined statement replaces the previously defined statement of the same number. A LEGEND statement without a number is treated as a LEGEND1 statement.

Cancel individual LEGEND statements by defining a LEGEND statement of the same number without options (a null statement):

```
legend4;
```

Canceling one LEGEND statement does not affect any other LEGEND definitions. To cancel all current LEGEND statements, use RESET= in a GOPTIONS statement:

```
goptions reset=legend;
```

Specifying RESET=GLOBAL or RESET=ALL cancels all current LEGEND definitions as well as other settings.

To display a list of current LEGEND definitions in the SAS LOG window, use the GOPTIONS procedure with the LEGEND option:

```
proc goptions legend nolist;
run;
```

Positioning the Legend

How to Position a Legend

By default, the legend shares the procedure output area with the procedure output, such as a map or bar chart. See "How Graphics Elements Are Placed in the Graphics Output Area" in *SAS/GRAPH: Reference*. However, several LEGEND statement options enable you to position a legend anywhere on the graphics output area and even to overlay the procedure output. This section describes these options and their effect on each other.

Positioning the Legend on the Graphics Output Area

There are two ways that you can position the legend on the graphics output area:

- Describe the general location of the legend with the POSITION= option. If necessary, fine-tune the position with the OFFSET= option.
- Position the legend explicitly with the ORIGIN=option.

Using POSITION= and OFFSET=

The values of the POSITION= option affect the legend in two ways:

- OUTSIDE and INSIDE determine whether the legend is located outside or inside the axis area.
- BOTTOM or MIDDLE or TOP (vertical position) and LEFT or CENTER or RIGHT (horizontal position) determine where the legend is located in relation to its OUTSIDE or INSIDE position.

Figure 3.5 on page 41 shows the legend positions inside the axis area.

Figure 3.5 *Legend Positions inside the Axis Area*

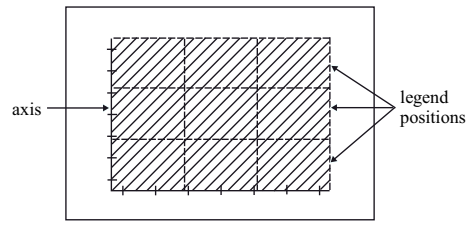
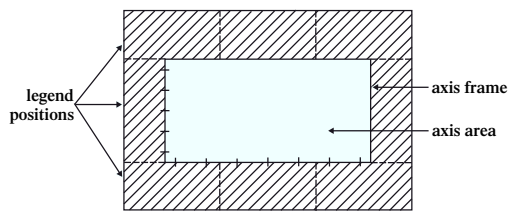


Figure 3.6 on page 41 shows legend positions outside the axis area.

Figure 3.6 *Legend Positions outside the Axis Area*



The default combination is POSITION=(BOTTOM CENTER OUTSIDE). The combination (OUTSIDE MIDDLE CENTER) is not valid.

Use OFFSET=(*x*,*y*) to adjust the position of the legend specified by the POSITION= option. The *x* value shifts the legend either left or right and the *y* value shifts the legend either up or down.

The offset values are always applied after the POSITION= request. For example, if POSITION=(TOP RIGHT OUTSIDE), the legend is located in the upper right corner of the graphics output area. If OFFSET=(0,0) is specified, the legend does not move. If OFFSET=(–5,–8)CM, the legend moves 5 centimeters to the left and 8 centimeters down.

Using ORIGIN=

Use ORIGIN=(*x*,*y*) to specify the coordinates of the exact location of the lower left corner of the legend box. Because ORIGIN=(0,0) is the lower left corner of the graphics output area, the values of *x* and *y* must be positive. If you specify negative values, a warning is issued and the default value is used.

Relating Legends to Other Graphics Elements

By default, the legend is inside the procedure output area and the space that it occupies reduces the size of the graph itself. If a choice must be made between displaying the legend or the graph, the legend is suppressed. To control how the legend relates to the other elements of the graph, use the MODE= option. These are values for the MODE= option:

- RESERVE reserve space for the legend outside the axis area and move the graph to make room for the legend. This is the default setting and is valid only when POSITION=OUTSIDE.
- PROTECT prevents the legend from being overwritten by the procedure output. PROTECT blanks out graphics elements, allowing only legend elements to be displayed in the legend's space.
- SHARE displays both graphics elements and legend elements in the same space. This setting is usually used when the legend is positioned inside the axis area. SHARE is useful when the graph has a space that the legend can fit into.

Interactions between POSITION= and MODE=

You cannot specify both POSITION=INSIDE and MODE=RESERVE because MODE=RESERVE assumes that the legend is outside the axis area, and POSITION=INSIDE positions the legend inside the axis area. Therefore, when you specify POSITION=INSIDE, change the value of the MODE= option to SHARE or PROTECT. Otherwise, SAS/GRAPH issues a warning and automatically changes the MODE= value to PROTECT.

Creating Drop Shadows and Block Effects

You can produce a drop shadow or a three-dimensional block effect behind the legend. Use the CSHADOW= or CBLOCK= option in the LEGEND statement in conjunction with the graphics option OFFSHADOW=(*x*,*y*).

The value of *x* determines how far the shadow or block extends to the right (positive numbers) or to the left (negative numbers) of the legend. The value of *y* determines how far the shadow or block extends above (positive numbers) or below (negative numbers) the legend. If OFFSHADOW=(0,0) is specified, the shadow or block is not visible.

By default, OFFSHADOW=(0.0625, -0.0625) IN. That is, the shadow or block extends 1/16th of an inch to the right and 1/16th of an inch below the legend.

Substituting BY Line Values in a Text String

The BY statement produces a BY line that contains the variable name and its value. If you specify the variable name, options are available to substitute the variable name and its value in text strings. To use the #BYVAR and #BYVAL options, insert the option in the text string at the position that you want the substitution text to appear. Both #BYVAR and #BYVAL specifications must be followed by a delimiting character. This can be either a space or other nonalphanumeric character, such as the quotation mark that ends the text string. If not, the specification is completely ignored and its text remains intact and is displayed with the rest of the string.

To allow a #BYVAR or #BYVAL substitution to be followed immediately by other text, with no delimiter, use a trailing dot (as with macro variables). The trailing dot is not displayed in the resolved text.

If you want a period to be displayed as the last character in the resolved text, use two dots after the #BYVAR or #BYVAL substitution.

The substitution for #BYVAL or #BYVAR does not occur if the following is true:

- The BY statement does not name the variable specified by #BYVAL or #BYVAR. For example, #BYVAL2 when there is only one BY variable or #BYVAL(ABC) when ABC is not a BY variable or does not exist.
- There is no BY statement at all.

When substitution does not occur, no error or warning message is issued and the option specification is displayed with the rest of the string. The graph continues to display a BY

line at the top of the page unless you suppress it by using the NOBYLINE option in an OPTION statement.

For more information, see the "BY Statement" on page 19.

Note: This feature is not available in the Annotate facility because BY lines are not created in a DATA step.

NOTE Statement

Writes lines of text in the output.

Type: Local

Restriction: Not supported by Java and ActiveX

Note: This local statement has limited use with the GMAP procedure. The specified note appears behind the map output by default. With some manipulation of coordinates, notes are useful around the perimeter or ocean area of a map.

See: "TITLE, FOOTNOTE, and NOTE Statements" on page 56

Syntax

NOTE <*text-arguments(s)*>;

ODS HTML Statement

Opens or closes the HTML destination.

Used by: GMAP

Requirement: On mainframes, either GPATH= or PATH= is required.

Syntax

ODS HTML <(<ID=> *identifier*)> <<*action*> | <option(s)>>;

Details

Description: ODS HTML Statement

This section describes the ODS HTML statement as it relates to SAS/GRAPH procedures. For complete information about the ODS HTML statement, see the *SAS Output Delivery System: User's Guide*.

The ODS HTML statement opens or closes the HTML destination. This destination is the default output destination for the Windows and UNIX operating systems. When the destination is open, the procedure produces output that is written in Hypertext Markup Language in the form of an HTML file. If no device is specified, SAS/GRAPH by default, creates a PNG file containing the graph. The HTML file references the PNG file in order to display the graph in a web page.

If DEVICE=JAVAMETA, graphics output is produced as metagraphics data. The browser passes the metacodes as a parameter to the Metaview applet. The Metaview applet renders the output defined by the metacodes, and displays the interactive graph in

a web page. For more information about DEVICE=JAVAMETA see "Developing Web Presentations for the Metaview Applet" in *SAS/GRAPH: Java Applets and ActiveX Control User's Guide*.

You can also use the DEVICE=JAVA and DEVICE=ACTIVEX options to create interactive graphics presentations for the web.

SAS/GRAPH adds data tip text to some graphs depending on the device specified. These data tips are generated by default using the values of fields in a SAS data set. You can specify the DESCRIPTION= option on the SAS/GRAPH procedure to change or remove the data tip text. For more information about using data tips see "Data Tips for Web Presentations" in *SAS/GRAPH: Reference*.

The FILE= option identifies the file that contains the HTML version of the procedure output. With SAS/GRAPH, the body file contains references to the graphs. If DEVICE=PNG, the graphs are stored in separate PNG files. When you view the body file in a browser, the graphs are automatically displayed. By default with ODS processing, both the HTML and PNG files are stored in the current WORK library. To specify a destination for all the HTML and PNG files, use the PATH= option. You can store the PNG files in a different location than the HTML files. Use the GPATH= option to specify a location for the PNG files, and the PATH= option to specify the location of the HTML files. In both cases, the destination must be an aggregate storage location.

Anchors

ODS HTML automatically creates an *anchor* for every piece of output generated by the SAS procedures. An anchor specifies a particular location within an HTML file. In SAS/GRAPH, an anchor usually defines a link target such as a graph whose location is defined in an IMG element.

In order for the links from the contents, page, or frame file to work, each piece of output in the body files must have a unique anchor to link to. The anchor for the first piece of output in a body file acts as the anchor for that file. These anchors are used by the frame and contents files, if they are created, to identify the targets for the links that ODS HTML automatically generates. For more information about using anchors with the ODS HTML statement, see the *SAS Output Delivery System: User's Guide*.

You can specify a name for an HTML anchor with:

```
ods html anchor="string";
```

This enables you to link directly to that identifying name.

PATTERN Statement

Defines the characteristics of patterns used in maps.

Used by: GMAP

Type: Global

Syntax

PATTERN<1 …255>
<COLOR=*pattern-color* | *_style_*>
<IMAGE=*fileref* | *"external-file"*>
<IMAGESTYLE=TILE | FIT>
<REPEAT=*number-of-times*>
<VALUE=*block-pattern* | *map-pattern* >;

Optional Arguments

COLOR=*pattern-color* | *_style_*

specifies the color of the fill. *Pattern-color* is any SAS/GRAPH color name. The _STYLE_ value specifies the appropriate color based on the current style. See "Using Colors in SAS/GRAPH Programs" in *SAS/GRAPH: Reference*.

Using the COLOR= option with a null value cancels the color specified in a previous PATTERN statement of the same number without affecting the values of other options.

The COLOR= option overrides the CPATTERN= graphics option.

The CFILL= option in the PIE and STAR statements overrides the COLOR= option. For details, see "Controlling Slice Patterns and Colors" in *SAS/GRAPH: Reference*.

No color can be specified for a PATTERN statement, that is, neither the COLOR= nor the CPATTERN= option is used. In this case the PATTERN statement rotates the specified fill through each color in the color list before the next PATTERN statement is used.

Alias	C=
Restriction	Partially supported by Java and ActiveX
Note	ActiveX assigns colors in a different order from Java, so the same data can appear differently with those two drivers.
See	"Working with PATTERN Statements" on page 52
Example	"Using BY-group Processing to Generate a Series of Charts" in *SAS/GRAPH: Reference*
CAUTION	**Omitting the COLOR= option in a PATTERN statement can cause the PATTERN statement to generate multiple PATTERN definitions.**

IMAGE=*fileref* | "*external-file*"

specifies an image file that is used to fill one or more areas of a map. The format of the external file specification varies across operating environments. See the companion reference for your specific operating environment.

Restriction	Partially supported by Java and ActiveX
Notes	When you specify an image file to fill a map, the map is not outlined. Also, the COLOR= and VALUE= options are ignored.
	For DEVICE=ACTIVEX and DEVICE=ACTXIMG, if you do not specify a pathname to the image, then the ActiveX control searches a predefined list of locations to try to find the image. If all else fails, the ActiveX control looks for the image on the web. It is recommended that you specify the pathname to the image.
See	"IMAGESTYLE=TILE	FIT" on page 45

IMAGESTYLE=TILE | FIT

specifies how the image specified in the IMAGE= option is to be applied to fill a map. The TILE value repeats the image as needed to fill the map. The FIT value stretches a single instance of the image to fill the map.

Default TILE

Restriction Partially supported by Java and ActiveX

REPEAT=*number-of-times*
specifies the number of times that a PATTERN definition is applied before the next PATTERN definition is used.

The behavior of the REPEAT= option depends on the color specification:

- If you use both the COLOR= and REPEAT= options in a PATTERN statement, the pattern is repeated the specified number of times in the specified color. The fill can be either the default solid or a fill specified with the VALUE= option.
- You can use the CPATTERN= option in a GOPTIONS statement to specify a single pattern color. Use the REPEAT= option either alone or with the VALUE= option in a PATTERN statement. The resulting hatch pattern is repeated the specified number of times.
- You can omit both the COLOR= and CPATTERN= options and use the REPEAT= option. If you use the REPEAT= option alone, it generates default solids. Or you can use the REPEAT= option in conjunction with the VALUE= option in a PATTERN statement. The resulting pattern is rotated through each color in the color list. The entire group generated by this cycle is then repeated the number of times specified in the REPEAT= option. Thus, the total number of patterns produced depends on the number of colors in the current color list.

Using REPEAT= with a null value cancels the repetition specified in a previous PATTERN statement of the same number without affecting the values of other options. Note that in most cases, it is preferable to use LEVELS=1 in the GMAP procedure rather than using this option in the PATTERN statement.

Alias R=

Default REPEAT=1

Restriction Partially supported by Java and ActiveX

See "Understanding Pattern Sequences" on page 54

VALUE=*block-pattern*
Patterns are specified for the blocks in block maps produced by the BLOCK statement in the GMAP procedure. (The map area from which the block rises takes a map pattern as described in the option "VALUE= *map-pattern*" on page 47.) See also "About Block Maps and Patterns" on page 234.

Values for *block-pattern* are as follows:

EMPTY
an empty pattern. Neither the Java applet nor the ActiveX control supports EMPTY.

Alias E

SOLID
a solid pattern (the only valid value for three-dimensional charts).

Alias S

style <density>

a shaded pattern. *Style* specifies the direction of the lines:

L

specifies left-slanting lines.

R

specifies right-slanting lines.

X

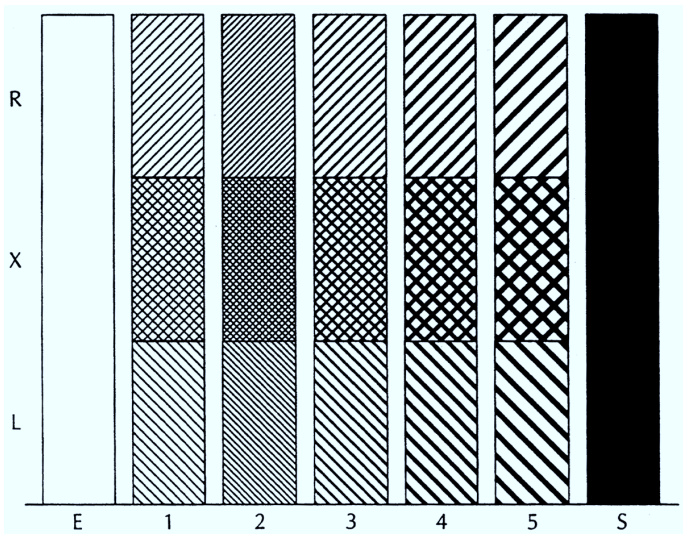
specifies crosshatched lines.

Density specifies the density of the pattern's shading. Specify a number from 1 to 5. The number 1 produces the lightest shading and 5 produces the heaviest shading.

Note: style<density> is not supported by the Java or ActiveX device drivers.

Figure 3.7 on page 47 shows all of the patterns available for bars and blocks.

Figure 3.7 *Bar and Block Patterns*



If no valid patterns are available, default bar and block fill patterns are selected in this order:

1. SOLID
2. X1–X5
3. L1–L5
4. R1–R5

Each fill is used once with every color in the color list unless a pattern color is specified. The entire sequence is repeated as many times as required to provide the necessary number of patterns.

Alias	V=
Restriction	Partially supported by Java and ActiveX

VALUE= ***map-pattern***

The VALUE= option specifies patterns for map area surfaces in block, choropleth, and prism maps produced by the BLOCK, CHORO, AND PRISM statements in the GMAP procedure.

Values for *map-pattern* are as follows:

MEMPTY
specifies an empty pattern.

Alias ME

Restriction EMPTY or E are not valid aliases when used with the map areas in block maps created by the GMAP procedure.

MSOLID
specifies a solid pattern.

Alias MS

Restriction SOLID or S are not valid aliases when used with the map areas in block maps created by the GMAP procedure.

M*density <style <angle> >*
specifies a shaded pattern.

Density specifies the density of the pattern's shading. Specify a number from 1 to 5. The number 1 produces the lightest shading and 5 produces the heaviest shading.

Style specifies the type of the pattern lines:

N
specifies parallel lines (the default).

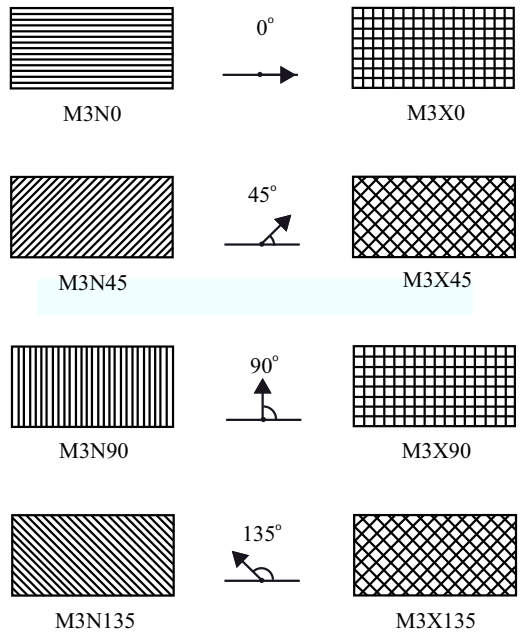
X
specifies crosshatched lines.

Angle specifies the angle of the pattern lines. Specify a number from 0 to 360. The number specifies the degrees at which the parallel lines are drawn, measured from the horizontal. By default, *angle* is 0 (lines are horizontal).

Note M*density<style<angle>>* is not supported by the Java or ActiveX device drivers.

Figure 3.8 on page 49 shows some typical map patterns.

Figure 3.8 *MapPatterns*



If no valid patterns are available, default plot fill patterns are selected in this order:

1. MSOLID
2. M2N0
3. M2N90
4. M2X45
5. M4N0
6. M4N90
7. M4X90

Each fill is used once with every color in the color list unless a pattern color is specified. The entire sequence is repeated as many times as required to provide the necessary number of patterns.

Alias	V=
Restriction	Partially supported by Java and ActiveX

Details

Description: PATTERN Statement

PATTERN statements create PATTERN definitions that define the color and type of area fill for patterns used in maps. This is the procedure and the graphics areas that it creates that use PATTERN definitions:

GMAP
: map areas in choropleth, block, and prism maps; blocks in block maps

In addition, certain Annotate facility functions and macros can use pattern specifications. and "Using Annotate Data Sets" in *SAS/GRAPH: Reference*.

You can use the PATTERN statement to control the fill and color of a pattern, and whether the pattern is repeated. There are various types of patterns:

- block patterns
- map patterns

Pattern fills can be solid or empty, or composed of parallel or crosshatched lines. In addition, you can specify device-dependent hardware patterns for polygon fills on devices that support hardware patterns.

If you do not create PATTERN definitions, SAS/GRAPH software generates them as needed and assigns them to your graphs by default. Generally, the default behavior is to rotate a solid pattern through the current color list. For details, see "About Default Patterns" on page 51.

Using the PATTERN Statement

How PATTERN Definitions Are Generated

PATTERN statements can be located anywhere in your SAS program. They are global and remain in effect until redefined, canceled, or until the end of your SAS session.

You can define up to 255 different PATTERN statements. A PATTERN statement without a number is treated as a PATTERN1 statement.

PATTERN statements generate one or more PATTERN definitions, depending on how the COLOR=, VALUE=, and IMAGE= options are used. For information about PATTERN definitions, see "Working with PATTERN Statements" on page 52, as well as the descriptions of " COLOR=*pattern-color* | *_style_*" on page 45, "IMAGE=*fileref* | "*external-file*"" on page 45, and "VALUE= *map-pattern*" on page 47.

PATTERN definitions are generated in the order in which the statements are numbered, regardless of gaps in the numbering or the statement's position in the program. Although it is common practice, you do not have to start with PATTERN1, and you do not have to use sequential statement numbers.

PATTERN definitions are applied automatically to all areas of the graphics output that require patterns. When assigning PATTERN definitions, SAS/GRAPH starts with the lowest-numbered definition with an appropriate fill specification or with no fill specification. It continues to use the specified patterns until all valid PATTERN definitions have been used. Then, if more patterns are required, SAS/GRAPH returns to the default pattern rotation, but continues to outline the areas in the same color as the fill.

Altering or Canceling PATTERN Statements

PATTERN statements are additive. You can define a PATTERN statement and later submit another PATTERN statement with the same number. The new PATTERN statement redefines or cancels only the options that are included in the new statement. Options not included in the new statement are not changed and remain in effect. For example, assume you define PATTERN4 as follows:

```
pattern4 value=x3 color=red repeat=2;
```

This statement cancels only REPEAT= without affecting the rest of the definition:

```
pattern4 repeat=;
```

Add or change options in the same way. This statement changes the color of the pattern from red to blue:

```
pattern4 color=blue;
```

After all these modifications, PATTERN4 has these characteristics:

```
pattern4 value=x3 color=blue;
```

Cancel individual PATTERN statements by defining a PATTERN statement of the same number without options (a null statement):

```
pattern4;
```

Canceling one PATTERN statement does not affect any other PATTERN definitions. To cancel all current PATTERN statements, use the RESET= option in a GOPTIONS statement:

```
goptions reset=pattern;
```

Specifying RESET=GLOBAL or RESET=ALL cancels all current PATTERN definitions as well as other settings.

To display a list of current PATTERN definitions in the LOG window, use the GOPTIONS procedure with the PATTERN option:

```
proc goptions pattern nolist;
run;
```

About Default Patterns

How SAS/GRAPH Generates and Assigns PATTERN Definitions

When a procedure produces a graph that needs one or more patterns, SAS/GRAPH does one of the following actions:

- automatically generates the appropriate default patterns and outlines to fill the areas
- uses patterns, colors, and outlines that are defined by PATTERN statements, graphics options, and procedure options.

In order to understand how SAS/GRAPH generates and assigns patterns defined with PATTERN statements, it is helpful to understand how it generates and assigns default patterns. The following sections describe the default pattern behavior for all procedures. See "Working with PATTERN Statements" on page 52 for details about defining patterns.

How Default Patterns and Outlines Are Generated

In general, the default pattern that SAS/GRAPH uses is a solid fill. The default colors are determined by the current style and the device.

SAS/GRAPH uses default patterns when no PATTERN statements are defined. The default colors are determined by the current style and the device.

Because the system option-GSTYLE-is in effect by default, the procedure uses the style's default block fill colors, widths, patterns, and outline colors when producing output. Specifically, SAS/GRAPH uses the default values when you do not specify any of the following:

- any PATTERN statements
- the CPATTERN= graphics option
- the COLORS= graphics options (that is, you use the device's default color list and it has more than one color)
- the COUTLINE= option in the action statement

If all of these conditions are true, then SAS/GRAPH performs the following operations:

- selects the first default fill for the appropriate pattern, which is always solid. It rotates once through the list of colors available in the current style, generating one

solid pattern for each color. If you use the default style colors and the first color in the list is either black or white, the procedure does not create a pattern in that color. If you specify a color list with the COLORS= graphics option, then the procedure uses all the colors in the list to generate the patterns.

Note: The one exception to the default solid pattern is the map area pattern in a block map produced by the GMAP procedure, which uses a hatch fill by default. By default the map areas and their outlines use the first color in the color list. This happens regardless of whether the list is the default device list or one specified with COLORS= in the GOPTIONS statement.

- uses the style's outline color to outline every patterned area.

If a procedure needs additional patterns, SAS/GRAPH selects the next default pattern fill appropriate to the graph and rotates it through the color list, skipping the foreground color as before. SAS/GRAPH continues in this fashion until it has generated enough patterns for the chart.

Things That Affect Default Patterns

Changing any of these conditions can change or override the default behavior:

- If you specify a color list with the COLORS= option in a GOPTIONS statement and the list contains more than one color, SAS/GRAPH rotates the default fills, beginning with SOLID, through that list. In this case, it uses every color, even if the foreground color is black (or white). The default outline color remains the foreground color.
- If you specify either COLORS=(*one-color*) or the CPATTERN= graphics option, the default fill changes from SOLID to the appropriate list of hatch patterns. SAS/GRAPH uses the specified color to generate one pattern definition for each hatch pattern in the list.

For a description of these options, see "Graphics Options and Device Parameters Dictionary " in *SAS/GRAPH: Reference*.

Working with PATTERN Statements

What You Can Specify with PATTERN Statements

With PATTERN statements, you can specify the following:

- the type of fill (VALUE=)
- the color of the fill (COLOR=)
- the images used to fill the bars in a two-dimensional chart (IMAGE=)
- how many times to apply the statement before using the next one (REPEAT=).

You can also use procedure options to specify the pattern outline color and the CPATTERN= graphics option to specify a default color for all patterns.

Whether you use PATTERN statement options alone or with each other affects the number and type of patterns your PATTERN statements generate. Depending on which options you use, you can explicitly specify every pattern used by your graphs. Or you can let the PATTERN statement generate a series of pattern definitions using either the color list or the list of default fills.

Explicitly Specifying Patterns

To explicitly specify all the patterns in your graph, you need to do one of the following for every pattern your graph requires:

- Provide a PATTERN statement that uses the COLOR= option to specify the pattern color, for example:

  ```
  pattern1 color=red;
  ```

  By default, the fill type is SOLID.

- Provide a PATTERN statement that uses both the COLOR= option and the VALUE= option to specify the fill, for example:

  ```
  pattern1 color=blue value=r3;
  ```

Including the COLOR= option in the PATTERN statement is the simplest way to assure that you get exactly the patterns that you want. When you use the COLOR= option, the PATTERN statement generates exactly one PATTERN definition for that statement. If you also use the REPEAT= option, the PATTERN definition is repeated the specified number of times.

Generating Multiple Pattern Definitions

You can also use PATTERN statements to generate multiple PATTERN definitions. To do this use the VALUE= option to specify the type of fill you want but omit the COLOR= option. For example:

```
pattern1 value=r3;
```

In this case, the PATTERN statement rotates the R3 fill through all the colors in the color list. For more information about pattern rotation, see "Understanding Pattern Sequences" on page 54.

Selecting an Appropriate Pattern

The type of fill you specify depends on the type of graph that you are producing:

Type of graph	Appropriate type of fill
block maps (PROC GMAP)	" VALUE=*block-pattern*" on page 46
map area surfaces (PROC GMAP)	"VALUE= *map-pattern*" on page 47

Note: You might specify a fill that is inappropriate for the type of graph that you are generating. For example, you might specify VALUE=L1 in a PATTERN statement for a choropleth map. In this case SAS/GRAPH ignores the PATTERN statement and continues searching for a valid pattern. If it does not find a definition with a valid fill specification, it uses default patterns instead.

Controlling Outline Colors

Whenever you use PATTERN statements, the default outline color uses the style's outline color to outline every patterned area.

You can change the outline color of any pattern, whether the pattern is default or user-defined. Use the COUTLINE= option or the CEMPTY= option, or both, in the action statement that generates the map.

The Effect of the CPATTERN= Graphics Option

Although the CPATTERN= graphics option is used most often with default patterns, it does affect the PATTERN statement. With default patterns (no PATTERN statements specified), it does the following:

- specifies the color for all patterns
- causes default patterns to use hatched fills instead of the default SOLID

In conjunction with the PATTERN statement that it does the following:

- With a PATTERN statement that specifies only a fill (VALUE=), the CPATTERN= option determines the color of that fill. For example, these statements produce two green, hatched patterns:

```
goptions cpattern=green;
pattern1 value=x3;
pattern2 value=x1;
```

- With a PATTERN statement that specifies only a color (COLOR=), the COLOR= option overrides the CPATTERN= color, but CPATTERN= causes the fill to be hatched, not the default SOLID. For example, these statements produce one red, hatched pattern:

```
goptions cpattern=green;
pattern1 color=red;
```

See also the description of "CPATTERN" in *SAS/GRAPH: Reference*.

Understanding Pattern Sequences

About Pattern Sequences

Pattern sequences are sets of PATTERN definitions that SAS/GRAPH automatically generates when a PATTERN statement specifies a fill but not a color. In this case, the specified fill is used once with every color in the color list. If the REPEAT= option is also used, the resulting PATTERN definitions are repeated the specified number of times.

Generating Pattern Sequences

SAS/GRAPH generates pattern sequences when a PATTERN statement uses VALUE= to specify a fill and all of the following conditions are also true:

- The COLOR= option is not used in the PATTERN statement.
- The CPATTERN= graphics option is not used.
- The color list, either default or user-specified, contains more than one color.

In this case, the PATTERN statement rotates the fill specified by the VALUE= option through every color in the color list. One PATTERN definition is generated for every color in the list. After every color has been used once, SAS/GRAPH goes to the next PATTERN statement. For example, suppose you specified the following color list and PATTERN statements for bar or block patterns:

```
goptions colors=(blue red green) ctext=black;
pattern1 color=red   value=x3;
pattern2 value=r3;
pattern3 color=blue  value=l3;
```

Here, **PATTERN1** generates the first PATTERN definition. **PATTERN2** omits the COLOR= option, so the specified fill is rotated through all three colors in the color list before the PATTERN3 statement is used. This table shows the color and fill of the PATTERN definitions that would be generated if nine patterns were required:

Definition Number	Source	Characteristics	
		Color	Fill
1	PATTERN1	Red	x3

Definition Number	Source	Characteristics	
		Color	Fill
2	PATTERN2	Blue	r3
3	PATTERN2	Red	r3
4	PATTERN2	Green	r3
5	PATTERN3	Blue	l3
6	First default	Blue	Solid
7	First default	Red	Solid
8	First default	Green	Solid
9	Second default	Blue	x1

Notice that after all the PATTERN statements are exhausted, the procedure begins using the default bar and block patterns, beginning with SOLID. Each fill from the default list is rotated through all three colors in the color list before the next default fill is used.

Repeating Pattern Sequences

If you use the REPEAT= option but not the COLOR= option, the sequence generated by cycling the definition through the color list is repeated the number of times specified by the REPEAT= option. For example, these statements illustrate the effect of the REPEAT= option on PATTERN statements both with and without explicit color specifications:

```
goptions colors=(red blue green);
pattern1 color=gold repeat=2;
pattern2 value=x1 repeat=2;
```

Here, **PATTERN1** is used twice and **PATTERN2** cycles through the list of three colors and then repeats this cycle a second time:

Sequence Number	Source	Characteristics	
		Color	Fill
1	PATTERN1	Gold	Solid (first default)
2	PATTERN1	Gold	Solid (first default)
3	PATTERN2	Red	x1
4	PATTERN2	Blue	x1
5	PATTERN2	Green	x1
6	PATTERN2	Red	x1

Sequence Number	Source	Characteristics	
		Color	Fill
7	PATTERN2	Blue	x1
8	PATTERN2	Green	x1

TITLE, FOOTNOTE, and NOTE Statements

Control the content, appearance, and placement of text.

Used by: GMAP

Type: TITLE and FOOTNOTE are global statements. NOTE is a local statement.

Restriction: The cumulative text from the TITLE1-10 statements must not exceed 256 characters for output using the SVG device.

Syntax

TITLE<1 …10> <*text-argument(s)*>;

FOOTNOTE<1 …10> <*text-argument(s)*>;

NOTE<*text-arguments(s)*>;

Summary of Optional Arguments

Appearance options

COLOR=*color*
: specifies the color for the following text, box, or line.

FONT=*font*
: specifies the font for the subsequent text.

HEIGHT=*text-height*<*units*>
: specifies the height of text characters in number of units.

Baseline angling and character rotation options

ANGLE=*degrees*
: specifies the angle of the baseline of the entire text string with respect to the horizontal.

LANGLE=*degrees*
: specifies the angle of the entire *text-string*(s).

ROTATE=*degrees*
: specifies the angle at which each character of text is rotated.

Boxing, underlining, and line drawing options

BCOLOR=*background-color*

BLANK=YES
: protects the box and its contents from being overwritten by any subsequent graphics elements.

BOX=1 …4
: draws a box around one line of text.

BSPACE=*box-space<units>*
: specifies the amount of space between the boxed text and the box.

DRAW=(*x,y …,x-n,y-n*)*<units>*
: draws lines anywhere on the graphics output area.

UNDERLIN=0 …3
: underlines subsequent text.

Linking option

ALT=*"text-string"*
: specifies descriptive text for a URL to which a title or footnote links, or the title or footnote itself.

LINK=*"URL"*
: specifies a uniform resource locator (URL) to which a title or footnote links.

Placement and spacing options

JUSTIFY=LEFT | CENTER | RIGHT
: specifies the alignment of the text string.

LSPACE=*line-space <units>*
: specifies the amount of spacing above and below lines of text.

MOVE=(*x,y*) *<units>*
: positions a text string.

WRAP
: wraps the text to a second line if the text does not fit on one line.

Text provision option

text-string(s)
: is an element of *text-argument(s)* and specifies one or more strings up to 200 characters. Any of the following *text-options* that are used to modify a *text-string* must precede *text-string*.

Optional Arguments

text-arguments(s) can be one or more of *"text-string"* and *text-options*. Text options must precede the *"text-string"* that they modify. *text-options* can be one or more of the following optional arguments, in any order:

ALT=*"text-string"*
: specifies descriptive text for a URL to which a title or footnote links, or the title or footnote itself. The *"text-string"* that you use to describe a title or a footnote, or the URL specified by the LINK= option, can contain occurrences of the variables #BYVAL, #BYVAR, and #BYLINE, as described in " *text-string(s)*" on page 69.

Supports The ALT= option can be used in conjunction with the LINK= option.

Note The title or footnote can display by using an ODS markup destination (such as HTML). Also, the corresponding ODS option NOGTITLE or NOGFOOTNOTE is specified. In this case, the title or footnote is rendered in the body of the HTML file rather than in the graphic itself. And the ALT= text is not associated with the title or footnote.

See "Controlling Where Titles and Footnotes Are Rendered" in *SAS/GRAPH: Reference*

" LINK="*URL*"" on page 67

Example "Example 11: Using Traditional Map Data to Produce a Drilldown Choropleth Map" on page 305

ANGLE=*degrees*

specifies the angle of the baseline of the entire text string with respect to the horizontal. A positive *degrees* value angles the *baseline* counterclockwise; a negative value angles it clockwise. By default, ANGLE=0 (horizontal).

Angled titles or footnotes might require more vertical space. Consequently, there might be an increase the size of the title area or the footnote area, thereby reducing the vertical space in the procedure output area.

Using the BOX= option with angled text does not produce angled boxes; the box is sized to accommodate the angled note.

Using the ANGLE= option after one text string and before another can reset some options to their default values. See "Using Options That Can Reset Other Options" on page 73.

The ANGLE= option has the same effect on the text as LANGLE=, except when you specify an angle of 90 degrees or –90 degrees. In these angle specifications, the procedure output area is shrunk from the left or right to accommodate the angled title or footnote. The result depends on the statement in which you use the option:

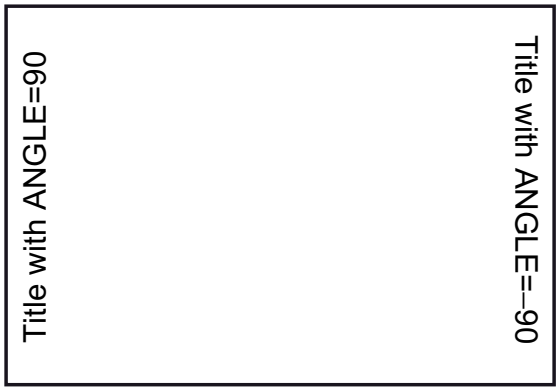
- With the TITLE statement:

  Figure 3.9 on page 58 shows how ANGLE=90-degrees or ANGLE=–90-degrees positions and rotates the title text.

ANGLE=90 positions the title at the left edge of the graphics output area, angled 90 -degrees (counterclockwise) and centered vertically.

ANGLE=–90 positions the title at the right edge of the graphics output area, angled –90-degrees (clockwise) and centered vertically.

Figure 3.9 *Positioning Titles with the ANGLE= Option*



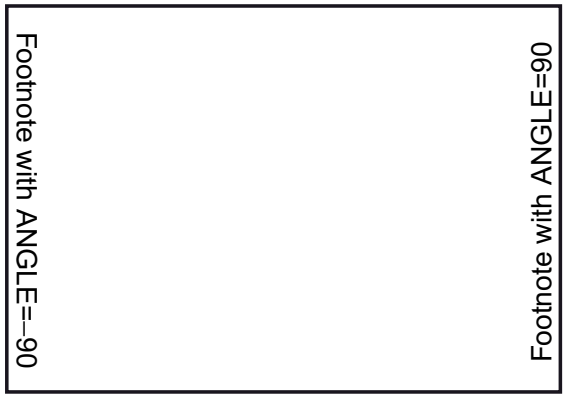
- With the FOOTNOTE statement:

  Figure 3.10 on page 59 shows how ANGLE=90 degrees or ANGLE=–90 degrees positions and rotates footnote text.

ANGLE=90 positions the footnote at the right edge of the graphics output area, angled 90 degrees (counterclockwise) and centered vertically.

ANGLE=–90 positions the footnote at the left edge of the graphics output area, angled –90 (clockwise) and centered vertically.

Figure 3.10 *Positioning Footnotes with the ANGLE=Option*



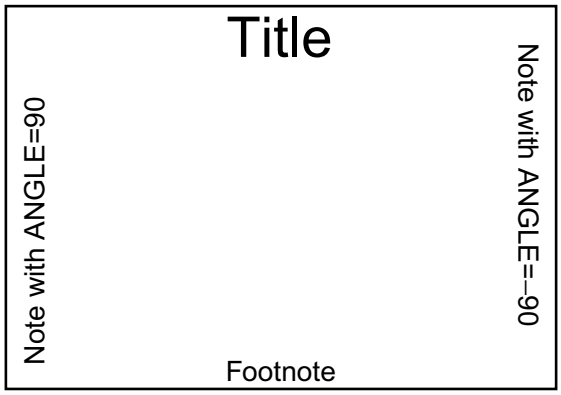
- With the NOTE statement:

  Figure 3.11 on page 59shows how ANGLE= 90 degrees or -90 degrees positions and rotates note text.

ANGLE=90 positions the note at the bottom of the left edge of the graphics output area, angled 90 degrees (counterclockwise) and reading from bottom to top.

ANGLE=–90 positions the note at the top of the right edge of the graphics output area, angled –90 (clockwise) and reading from top to bottom.

Figure 3.11 *Positioning Notes with the ANGLE= Option*



Alias	A=
Restriction	Not supported by Java and ActiveX
See	" LANGLE=*degrees*" on page 65
	" ROTATE=*degrees*" on page 68
Example	"Example 1: Enhancing Titles" on page 74

BCOLOR=*background-color*

specifies the background color of a box produced by the BOX= option. By default, the background color of the box is the same as the background color for the entire graph. The color of the frame of the box is determined by the color specification used in BOX=.

Alias	BC=
Restriction	If you omit BOX=, BCOLOR= is ignored.
Note	The BCOLOR= option can be reset by the ANGLE= or JUSTIFY= options, or by the MOVE= option with absolute coordinates.
See	"Using Options That Can Reset Other Options" on page 73 for details
	" BOX=1 …4 " on page 60
Example	"Example 1: Enhancing Titles" on page 74

BLANK=YES

protects the box and its contents from being overwritten by any subsequent graphics elements. It does this by blanking out the area where the box is displayed. The BLANK= option enables you to overlay graphics elements with boxed text. It is ignored if you omit the BOX= option. Because titles and footnotes are written from the highest numbered to the lowest numbered, the BLANK= option only blanks out titles and footnotes of a lower number.

Alias	BL=
Restriction	Not supported by Java and ActiveX
Note	The BLANK= option can be reset by the ANGLE= or JUSTIFY= options, or by the MOVE= option with absolute coordinates.
See	"Using Options That Can Reset Other Options" on page 73 for details
	" BOX=1 …4 " on page 60
Example	"Example 1: Enhancing Titles" on page 74

BOX=1 …4

draws a box around one line of text. A value of 1 produces the thinnest box lines; 4 produces the thickest. Boxing angled text does not produce an angled box; the box is sized to include the angled text.

The color of the box is one of the following:

- the color specified by the COLOR= option in the statement
- the default text color

The COLOR= option, preceding the BOX= option, controls the box frame color. To color the background of the box, use the BCOLOR= option. Specify the same color with both the COLOR= option and the BCOLOR= option to effect a box without a frame.

You can include more than one text string in the box as long as no text break occurs between the strings. That is, you cannot use the JUSTIFY= option to create multiple lines of text within a box.

To draw a box around multiple lines of text, you can do either of the following:

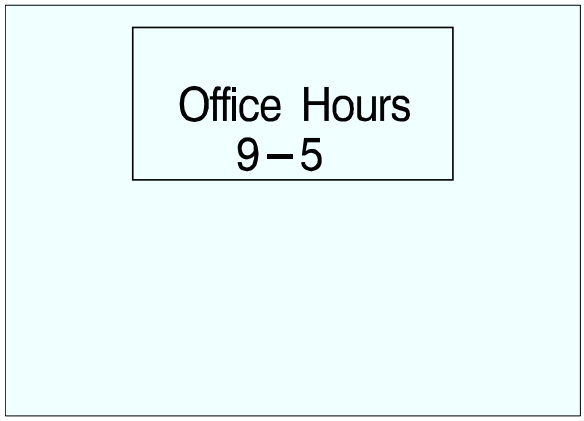
- Use the MOVE= option with relative coordinates to position an additional line of text relative to the preceding line and enclose them within the box drawn by the BOX= option. For example, this statement produces the boxed note shown in Figure 3.12 on page 61:

```
note font=swiss justify=center box=1
   "Office Hours"      move=(-7,-2)pct "9-5";
```

  The MOVE=option in this NOTE statement overrides the JUSTIFY= option that center-positioned the first line of text. The MOVE= option specifies the *x* and *y* coordinates in percentage units. The text string '9–5' is positioned relative to the first text string 'Office Hours'.

- Use the DRAW= option to draw the box and do not use the BOX= option.

Figure 3.12 *Using the BOX= Option and the MOVE= Option to Box Multiple Lines of Text*



Alias	BO=
Restriction	Not supported by Java and ActiveX
Note	The BOX= option can be reset by the ANGLE= or JUSTIFY= options, or by the MOVE= options with absolute coordinates.
See	"Using Options That Can Reset Other Options" on page 73 for details
	" BCOLOR=*background-color*" on page 60
	" BLANK=YES " on page 60
	" BSPACE=*box-space<units>*" on page 61
Example	"Example 1: Enhancing Titles" on page 74

BSPACE=*box-space<units>*

specifies the amount of space between the boxed text and the box. The space above the text is measured from the font maximum, and the space below the text is measured from the font minimum. By default, BSPACE=1.

The spacing is uniform around the box. Notice that BSPACE=.5IN leaves one-half inch of space between the text and the top, bottom, and sides of the box.

Alias	BS=
Restriction	Not supported by Java and ActiveX

Interaction If the BOX= option is not used, the BSPACE= option is ignored

Note The BSPACE= option can be reset by the ANGLE= or JUSTIFY= options, or by the MOVE= option with absolute coordinates.

See "Using Options That Can Reset Other Options" on page 73 for details

" BOX=1 …4 " on page 60

COLOR=*color*

specifies the color for the following text, box, or line. The COLOR= option affects all text, lines, and boxes that follow it and stays in effect until another COLOR= specification is encountered.

Change colors as often as you like. For example, this statement produces a title with red text in a box with a blue frame and a cream background:

```
title color=red "Total Sales" color=blue
  box=3 bcolor=cream;
```

Although the BCOLOR= option controls the background color of the box, the frame color is controlled with the COLOR= option that precedes the BOX= option.

If you omit the COLOR= option, a color specification is searched for in this order:

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

Alias C=

Style reference Color attribute of the GraphTitle1Text (TITLE1) and the GraphTitleText (TITLE2...n) elements

See " BCOLOR=*background-color*" on page 60

"Controlling Titles and Footnotes" in *SAS/GRAPH: Reference*

DRAW=(*x,y …,x-n,y-n*)<*units*>

draws lines anywhere on the graphics output area. Lines are drawn using *x* and *y* as absolute or relative coordinates. The following table shows the specifications for absolute and relative coordinates:

Table 3.1 *Coordinate Specifications*

Absolute Coordinates	Relative Coordinates
x<*units*>	±*x*<*units*>
y<*units*>	±*y*<*units*>

The coordinate position (0,0) is the lower left corner of the graphics output area. Specify at least two coordinate pairs. Commas between coordinates are optional; blanks can be used instead. The DRAW= option does not affect the positioning of text.

The starting point for lines specified with relative coordinates begins at the end of the most recently drawn text or line in the current statement. If no text or line has been drawn in the current statement, a warning is issued. The relative draw is measured from where a zero-length text string would have ended, given the normal placement for the statement.

You can mix relative and absolute coordinates. For example, DRAW=(+0,+0,+0,1IN) draws a vertical line from the end of the text to one inch from the bottom of the graphics output area.

Alias	D=
Restriction	Not supported by Java and ActiveX
Example	"Example 1: Enhancing Titles" on page 74

FONT=*font*

specifies the font for the subsequent text. See "Specifying Fonts in SAS/GRAPH Programs" in *SAS/GRAPH: Reference* for details about specifying SAS/GRAPH fonts. If you omit this option, a font specification is searched for in this order:

- for a TITLE1 statement:
  1. the FTITLE= option in a GOPTIONS statement
  2. the FTEXT= option in a GOPTIONS statement
  3. the default font, SWISS (COMPLEX in Release 6.06 and earlier)
- for all other TITLE statements and the FOOTNOTE and NOTE statements:
  1. the FTEXT= option in a GOPTIONS statement
  2. the default hardware font, NONE

Alias	F=
Style reference	Font attribute of the GraphTitle1Text (TITLE1) and the GraphTitleText (TITLE2...n) elements
Notes	Font names greater than eight characters in length must be enclosed in quotation marks.
	If the TITLE or FOOTNOTE is displayed using an ODS markup destination and the corresponding NOGTITLE or NOGFOOTNOTE option is specified, then the **`bold`** and **`italic`** FONT attributes are on by default. However, if you specify different attributes with the FONT= option, the **`bold`** and **`italic`** attributes are turned off.
See	"Controlling Titles and Footnotes" in *SAS/GRAPH: Reference*
Example	"Example 1: Enhancing Titles" on page 74

HEIGHT=*text-height<units>*

specifies the height of text characters in number of units.

Height is measured from the font minimum to the capline. Ascenders can extend above the capline, depending on the font.

If your text line is too long to be displayed in the height specified in the HEIGHT= option, the height specification is reduced so that the text can be displayed. A note in the SAS log tells you what percentage of the specified size was used.

If you omit the HEIGHT= option, a text height specification is searched for in this order:

- for a TITLE or TITLE1 statement:
  1. the HTITLE= option in a GOPTIONS statement
  2. the HTEXT= option in a GOPTIONS statement
  3. the default value specified by the current ODS style.
- for all other TITLE statements and the FOOTNOTE and NOTE statements:
  1. the HTEXT= option in a GOPTIONS statement
  2. the default value, 1

Alias	H=
Default	For TITLE or TITLE1 statements, HEIGHT is determined by the current ODS style. HEIGHT=1 for subsequent TITLE statements.
Style reference	Height attribute of the GraphTitle1Text (TITLE1) and the GraphTitleText (TITLE2...n) elements
Restriction	Partially supported by Java and ActiveX
Notes	The Java applet and ActiveX control enable you to control the relative height of text with the HEIGHT= option, but not the absolute height in terms of specific units.
	A TITLE or FOOTNOTE statement without a number is treated as a TITLE1 or FOOTNOTE1 statement.
See	"Controlling Titles and Footnotes" in *SAS/GRAPH: Reference*
Examples	
	"Example 1: Enhancing Titles" on page 74

JUSTIFY=LEFT | CENTER | RIGHT

specifies the alignment of the text string. The default depends on the statement with which you use the JUSTIFY= option:

- for a FOOTNOTE statement the default is CENTER
- for a NOTE statement the default is LEFT
- for a TITLE statement the default is CENTER.

All the text strings following JUSTIFY= are treated as a single string and are displayed as one line that is left-, right-, or center-aligned.

You can change the justification within a single line of text. For example, this NOTE statement displays a date on the left side of the output and the page number on the same line on the right:

```
note "June 28, 1997" justify=right "Page 3";
```

In addition, you can use the JUSTIFY= option to produce multiple lines of text by repeating the JUSTIFY= option with the same value before the text string for each line. Multiple lines of text with the same justification are blocked together. For example, this TITLE statement produces a three-line title with each line right-justified:

```
title justify=right "First Line"
      justify=right "Second Line"
      justify=right "Third Line";
```

You can get the same effect with three TITLE statements, each specifying JUSTIFY=RIGHT. You can produce a block of text by specifying the same justification for multiple text strings. You can then change the justification for an additional text string. The result is that the text is placed on the same line as the first string specified in the statement.

Alias J=L | C | R

Note Using the JUSTIFY= option after one text string and before another can reset some options to their default values.

See "Using Options That Can Reset Other Options" on page 73 for details

Example "Rotating Plot Symbols through the Color List" in *SAS/GRAPH: Reference*

LANGLE=*degrees*

specifies the angle of the *baseline* of the entire text string(s) with respect to the horizontal A positive value for *degrees* moves the *baseline* counterclockwise; a negative value moves it clockwise. By default, LANGLE=0 (horizontal).

Angled titles or footnotes might require more vertical space. Consequently, there might be an increase the size of the title area or the footnote area, thereby reducing the vertical space in the procedure output area.

Using the BOX= option with angled text does not produce an angled box; the box is sized to accommodate the angled note.

Unlike the ANGLE= option, the LANGLE= option does not reset any other options. Therefore, the LANGLE= option is easier to use because you do not need to repeat options after a text break.

The LANGLE= option has the same effect on the text as the ANGLE= option, except when an angle of 90 degrees or –90 degrees is specified. The result depends on the statement in which you use the option:

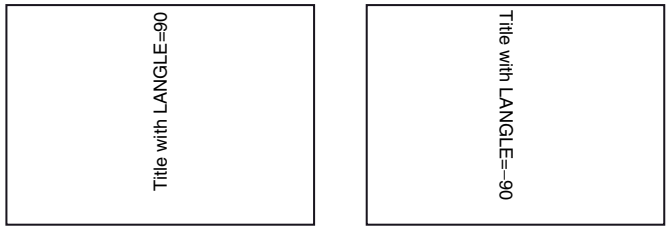
- With the TITLE statement:

  Figure 3.13 on page 66 shows how LANGLE=90 degrees and LANGLE=–90 degrees positions and rotates titles.

LANGLE=90 angles the title 90 degrees (counterclockwise) so that it reads from bottom to top. The title is centered horizontally and positioned at the top of the picture.

LANGLE=-90 angles the title –90 degrees (clockwise) so that it reads from top to bottom. The title is centered horizontally and positioned at the top of the picture.

Figure 3.13 *Positioning Titles with the LANGLE= Option*



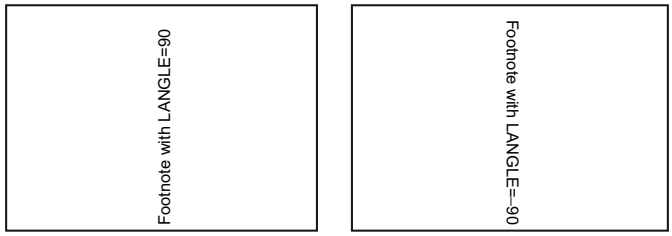
- With the FOOTNOTE statement:

  Figure 3.14 on page 66 shows how LANGLE=90 degrees and LANGLE=–90 degrees positions and rotates footnotes.

LANGLE=90 angles the footnote 90 degrees (counterclockwise) so that it reads from bottom to top. The footnote is centered horizontally and positioned as the bottom of the picture.

LANGLE=–90 angles the footnote –90 degrees (clockwise) so that it reads from top to bottom. The footnote is centered horizontally and positioned at the bottom of the picture.

Figure 3.14 *Positioning Footnotes with the LANGLE= Option*



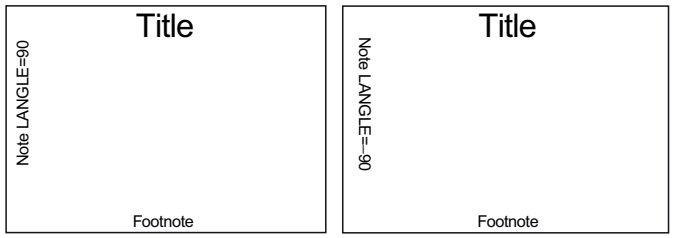
- With the NOTE statement:

  Figure 3.15 on page 66 shows how LANGLE=90 degrees and LANGLE=–90 degrees positions and rotates notes.

LANGLE=90 positions the note at the top of the left edge of the procedure output area, angled 90 degrees (counterclockwise) so that it reads from bottom to top.

LANGLE=–90 positions the note at the top of the left edge of the procedure output area, angled –90 degrees (clockwise) so that it reads from top to bottom.

Figure 3.15 *Positioning Notes with the LANGLE= Option*



Alias LA=

Restriction Not supported by Java and ActiveX

See " ANGLE=*degrees*" on page 58

LINK=*"URL"*

specifies a uniform resource locator (URL) to which a title or footnote links. The *text-string* that you use to specify the URL can contain occurrences of the variables #BYVAL, #BYVAR, and #BYLINE, as described in " *text-string(s)*" on page 69.

Supports The LINK= option can be used in conjunction with the ALT= option.

Note The title or footnote can display using an ODS markup destination (such as HTML). And the corresponding ODS option NOGTITLE or NOGFOOTNOTE can be specified. Then the title or footnote is rendered in the body of the HTML file rather than in the graphic itself. Specifying the NOGTITLE or NOGFOOTNOTE options results in increasing the amount of space allowed for the procedure output area, which can result in increasing the size of the graph. Space that would have been used for the title or footnote is devoted instead to the graph. You might need to be aware of this possible difference if you are using annotate or map coordinates.

See "Controlling Where Titles and Footnotes Are Rendered" in *SAS/GRAPH: Reference*

" ALT=*"text-string"*" on page 57

Example "Example 11: Using Traditional Map Data to Produce a Drilldown Choropleth Map" on page 305

LSPACE=*line-space <units>*

specifies the amount of spacing above lines of note and title text and the amount of spacing below lines of footnote text. For notes and titles, the spacing is measured from the capline of the current line to the font minimum of the line above. For footnotes, the spacing is measured from the font minimum of the current line to the capline of the line below.

Alias LS=

Default LSPACE=0

Restriction Not supported by Java and ActiveX

Note The LSPACE= option can be reset by the ANGLE= or JUSTIFY= option, or by the MOVE= option with absolute coordinates.

See "Using Options That Can Reset Other Options" on page 73 for details

MOVE=(*x*,*y*) *<units>*

positions an initial or subsequent text string, or lines of text anywhere on the graphics output area. The positioning uses *x* and *y* as absolute or relative coordinates. The following table shows the specifications for absolute and relative coordinates:

Table 3.2 *Coordinate Specifications*

Absolute Coordinates	Relative Coordinates
x<units>	*±x<units>*

Absolute Coordinates	Relative Coordinates
y<units>	±*y<units>*

Commas between coordinates are optional; you can use blanks instead.

The starting point for lines specified with absolute coordinates is at the bottom left corner of the graphics output area. For example, the following code fragment uses the NOTE statement to add a simple line of text to the upper right quadrant of the graphics output area:

```
note move=(65,70)pct color=purple "My Note";
```

The starting point for lines specified with relative coordinates is at the end of the most recently drawn text or line in the current statement. If no text or line has been drawn in the current statement, a warning is issued in the SAS log. The relative move is measured from where a zero-length text string would have ended, given the normal placement for the statement.

You can mix relative and absolute coordinates.

Alias M=

Restriction Not supported by Java and ActiveX

Interactions The MOVE= option overrides a JUSTIFY= option specified for the same text string

A NOTE, FOOTNOTE, or TITLE statement can use the MOVE= option to position the text so that the statement does not use its default position. In this case, the text of the next NOTE, FOOTNOTE, or TITLE statement occupies the unused position and no blank lines are displayed

Notes You can specify the MOVE= option with at least one absolute coordinate. If the option follows one text string and precedes another, some options can be reset to their default values.

If you specify the GUNIT graphics option, then that unit is the default unit. If you do not specify the GUNIT= graphics option, then the default unit is CELLS.

See "Using Options That Can Reset Other Options" on page 73 for details

" BOX=1 …4 " on page 60 for illustrative code that draws a box around lines of text positioned with relative coordinates

Examples

"Example 1: Enhancing Titles" on page 74

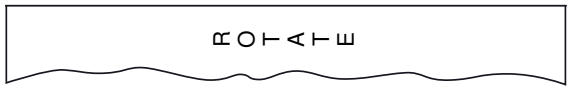
ROTATE=*degrees*

specifies the angle at which each character of text is rotated with respect to the *baseline* of the text string. The angle is measured from the current text *baseline* angle, which is specified by the ANGLE= or LANGLE= options. By default, the *baseline* is horizontal. A positive value for *degrees* rotates the character

counterclockwise; a negative value rotates it clockwise. By default, ROTATE=0 (parallel to the *baseline*).

Figure 3.16 on page 69 shows how characters are positioned when ROTATE=90 is used with the default (horizontal) *baseline*.

Figure 3.16 *Tilting Characters with the ROTATE= Option*



Alias	R=
Restriction	Not supported by Java and ActiveX
See	" ANGLE=*degrees*" on page 58
Example	"Example 1: Enhancing Titles" on page 74

text-string(s)

is an element of *text-argument(s)* and specifies one or more strings up to 200 characters. Any of the following *text-options* that are used to modify a *text-string* must precede *text-string*. You must enclose text strings in single or double quotation marks. The text appears exactly as you enter it in the statement, including uppercase and lowercase characters and blanks.

To use single quotation marks or apostrophes within the title, you can do either of the following:

- use a pair of single quotation marks together:

```
footnote 'All''s Well That Ends Well';
```

- enclose the text in double quotation marks:

```
footnote "All's Well That Ends Well";
```

Because FOOTNOTE, NOTE, and TITLE statements concatenate all text strings, the strings must contain the correct spacing. With a series of strings, add blanks at the beginning of a text string rather than at the end, as in this example:

```
note color=red "Sales:" color=blue " 2000";
```

With some fonts, you produce certain characters by specifying a hexadecimal value in quotation marks. A trailing **x** identifies a string as a hexadecimal value. For example, this statement[1] produces the title **Profits Increase 3,000**:

```
title font=swiss "Profits Increase " "9c"x "3,000";
```

For more information see "Specifying Special Characters Using Character and Hexadecimal Codes" in *SAS/GRAPH: Reference*.

In addition, if you have a BY statement and you specify the variable that it names, you can embed one or both of the following in the string:

#BYLINE

substitutes the entire BY line without leading or trailing blanks for #BYLINE in the text string. It also displays the BY line in the footnote, note, or title produced by the statement.

1 This statement assumes you are using a U.S. key map.

#BYVAL*n* | #BYVAL(*BY-variable-name*)
substitutes the current value of the specified BY variable for #BYVAL in the text string and displays the value produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAL should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAL2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAL(YEAR) specifies the BY variable, YEAR. *Variable-name* is not case sensitive.

Examples

"Combining Graphs and Reports in a Web Page" in *SAS/GRAPH: Reference*

#BYVAR*n* | #BYVAR(*BY-variable-name*)
substitutes the name of the BY variable or label associated with the variable (whatever the BY line would normally display) for #BYVAR in the text string and displays the name or label produced by the statement. Specify the variable with one of these:

n
specifies which variable in the BY statement #BYVAR should use. The value of *n* indicates the position of the variable in the BY statement. For example, #BYVAR2 specifies the second variable in the BY statement.

BY-variable-name
names the BY variable. For example, #BYVAR(SITES) specifies the BY variable, SITES. *Variable-name* is not case sensitive.

Note A BY variable name displayed is always in uppercase. If a label is used, it appears as specified in the LABEL statement.

See "Substituting BY Line Values in a Text String" on page 73

Restriction #BYVAL or #BYVAR substitution in a text string is not available in the Annotate facility. The reason is that BY lines are not created in a DATA step.

UNDERLIN=0 …3
underlines subsequent text. Values of 1, 2 and 3 underline with an increasingly thicker line. UNDERLIN=0 halts underlining for subsequent text.

Underlines follow the text *baseline*. If you use an LANGLE= or ANGLE= option for the line of text, the underline is drawn at the same angle as the text. Underlines do not break up to follow rotated characters.

To make the text and the underline the same color, specify a COLOR= option before the UNDERLIN= option that precedes the text string. To make the text a different color, specify the COLOR= option after the UNDERLIN= option.

Aliases U=

UNDERLINE=

Restriction Partially supported by Java and ActiveX

Notes The UNDERLIN= option can be reset by the ANGLE= or JUSTIFY= option, or by the MOVE= option with absolute coordinates. See "Using Options That Can Reset Other Options" on page 73 for details.

The Java applet and ActiveX control underline text when the UNDERLIN= option is specified, but they do not vary the thickness of the line.

See " ROTATE=*degrees*" on page 68

Example "Example 1: Enhancing Titles" on page 74

WRAP

wraps the text to a second line if the text does not fit on one line. If the WRAP option is omitted, the text font-size is reduced until the text fits on one line. Wrapping occurs at the last blank before the text meets the end of the window. If there are no blanks in the text string, then there is no wrapping.

Restriction The WRAP option does not work with the BOX, BLANK, UNDERLINE, and MOVE options.

Details

Using TITLE and FOOTNOTE Statements

You can define TITLE and FOOTNOTE statements anywhere in your SAS program. They are global and remain in effect until you cancel them or until you end your SAS session. All currently defined FOOTNOTE and TITLE statements are automatically displayed.

You can define up to ten TITLE statements and ten FOOTNOTE statements in your SAS session. A TITLE or FOOTNOTE statement without a number is treated as a TITLE1 or FOOTNOTE1 statement. You do not have to start with TITLE1 and you do not have to use sequential statement numbers. Skipping a number in the sequence leaves a blank line.

You can use as many text strings and options as you want, but place the options before the text strings that they modify. See "Using Multiple Options" on page 72.

The most recently specified TITLE or FOOTNOTE statement of any number completely replaces any other TITLE or FOOTNOTE statement of that number. In addition, it cancels all TITLE or FOOTNOTE statements of a higher number. For example, if you define TITLE1, TITLE2, and TITLE3, then resubmitting the TITLE2 statement cancels TITLE3.

To cancel individual TITLE or FOOTNOTE statements, define a TITLE or FOOTNOTE statement of the same number without options (a null statement):

```
title4;
```

But remember that this cancels all other existing statements of a higher number.

To cancel all current TITLE or FOOTNOTE statements, use the RESET= graphics option in a GOPTIONS statement:

```
goptions reset=footnote;
```

Specifying RESET=GLOBAL or RESET=ALL also cancels all current TITLE and FOOTNOTE statements as well as other settings.

Note: The cumulative text from the TITLE1-10 statements must not exceed 256 characters for output using the SVG device.

Using the NOTE Statement

NOTE statements are local, not global, and they must be defined within a procedure or RUN-group with which they are used. They remain in effect for the duration of the procedure that includes NOTE statements in any of its RUN-groups or until you end your SAS session. All notes defined in the current RUN group, as well as those defined in previous RUN-groups, are displayed in the output as long as the procedure remains active.

You can use as many text strings and options as you want, but place the options before the text strings that they modify. See "Using Multiple Options" on page 72.

Using Multiple Options

In each statement, you can use as many text strings and options as you want, but you must place the options before the text strings that they modify. Most options affect all text strings that follow them in the same statement, unless the option is explicitly reset to another value. In general, TITLE, FOOTNOTE, and NOTE statement options stay in effect until one of these events occurs:

- The end of the statement is reached.
- A new specification is made for that option.

For example, this statement specifies that one part of the note is red and another part is blue, but the height for all of the text is 4:

```
note height=4 color=red "Red Tide"
     color=blue " Effects on Coastal Fishing";
```

Controlling Placement of Text

The following options enable the positioning of text on the graphics output area:

- " JUSTIFY=LEFT | CENTER | RIGHT" on page 64 aligns the text string
- " MOVE=(*x*,*y*) <*units*>" on page 67 positions a line or lines of text anywhere on the graphics output area using *x* and *y* as absolute or relative coordinates.

  Figure 3.12 on page 61 illustrates using the MOVE= option with relative coordinates to position a subsequent line of text
- "WRAP" on page 71 wraps the text to a second line

See the description of each option for usage details and restrictions.

Setting Defaults

You can set default characteristics for titles (including TITLE1 definitions), footnotes, and notes by using the following graphics options in a GOPTIONS statement:

CTITLE=*color*
: sets the default color for all titles, footnotes, and notes; overridden by the COLOR= option in a TITLE, FOOTNOTE, or NOTE statement.

CTEXT=*text-color*
: sets the default color for all text; overridden by the CTITLE= option for titles, footnotes, and notes.

FTITLE=*title-font*
sets the default font for TITLE1 definitions; overridden by the FONT= option in the TITLE1 statement.

FTEXT=*text-font*
sets the default font for all text, including the TITLE1 statement if the FTITLE= option is not used; overridden by the FONT= option a TITLE, FOOTNOTE, or NOTE statement.

HTITLE=*height*<*units*>
sets the default height for TITLE1 definitions; overridden by the HEIGHT= option in the TITLE1 statement.

HTEXT=*n*<*units*>
sets the default height for all text, including the TITLE1 statement if the HTITLE= option is not used; overridden by the HEIGHT= option a TITLE, FOOTNOTE, or NOTE statement.

See "Graphics Options and Device Parameters Dictionary " in *SAS/GRAPH: Reference* for a complete description of each option.

Using Options That Can Reset Other Options

The ANGLE=, MOVE=, and JUSTIFY= options affect the position of the text and cause text breaks. (To cause a text break, the MOVE= option must have at least one absolute coordinate.) When a statement contains multiple text strings, the resulting text break can cause the following options to reset to their default values:

- BCOLOR=
- BLANK=
- BOX=
- BSPACE=
- LSPACE=
- UNDERLIN=.

Note: The LANGLE= option does not cause a text break.

In a TITLE, FOOTNOTE, or NOTE statement, before the first text string, you can use an option that can be reset (such as the UNDERLIN= option). Before the second string, if you use an option that resets it (such as the JUSTIFY= option), the first option does not affect the second string. In order for the first option to affect the second string, repeat the option and position it after the resetting option and before the text string.

For example, this statement produces a two-line title in which only the first line is underlined:

```
title underlin=2 "Line 1" justify=left "Line 2";
```

To underline Line 2, repeat the UNDERLIN= option before the second text string and after the JUSTIFY= option:

```
title underlin=2 "Line 1" justify=left
      underlin=2 "Line 2";
```

Substituting BY Line Values in a Text String

The BY statement produces a BY line that contains the variable name and its value. If you specify the variable name, options are available to substitute the variable name and its value in text strings. To use the #BYVAR and #BYVAL options, insert the option in the text string at the position that you want the substitution text to appear. Both

#BYVAR and #BYVAL specifications must be followed by a delimiting character. This can be either a space or other nonalphanumeric character, such as the quotation mark that ends the text string. If not, the specification is completely ignored and its text remains intact and is displayed with the rest of the string.

To allow a #BYVAR or #BYVAL substitution to be followed immediately by other text, with no delimiter, use a trailing dot (as with macro variables). The trailing dot is not displayed in the resolved text.

If you want a period to be displayed as the last character in the resolved text, use two dots after the #BYVAR or #BYVAL substitution.

The substitution for #BYVAL or #BYVAR does not occur if the following is true:

- The BY statement does not name the variable specified by #BYVAL or #BYVAR. For example, #BYVAL2 when there is only one BY variable or #BYVAL(ABC) when ABC is not a BY variable or does not exist.
- There is no BY statement at all.

When substitution does not occur, no error or warning message is issued and the option specification is displayed with the rest of the string. The graph continues to display a BY line at the top of the page unless you suppress it by using the NOBYLINE option in an OPTION statement.

For more information, see the "BY Statement" on page 19.

Note: This feature is not available in the Annotate facility because BY lines are not created in a DATA step.

Examples

Example 1: Enhancing Titles

Features: GOPTIONS statement options

BORDER

TITLE statement options

ANGLE=
BCOLOR=
BLANK=
BOX=
COLOR=
DRAW=
FONT=
HEIGHT=
MOVE=
ROTATE=
"*text-string*"
UNDERLIN=

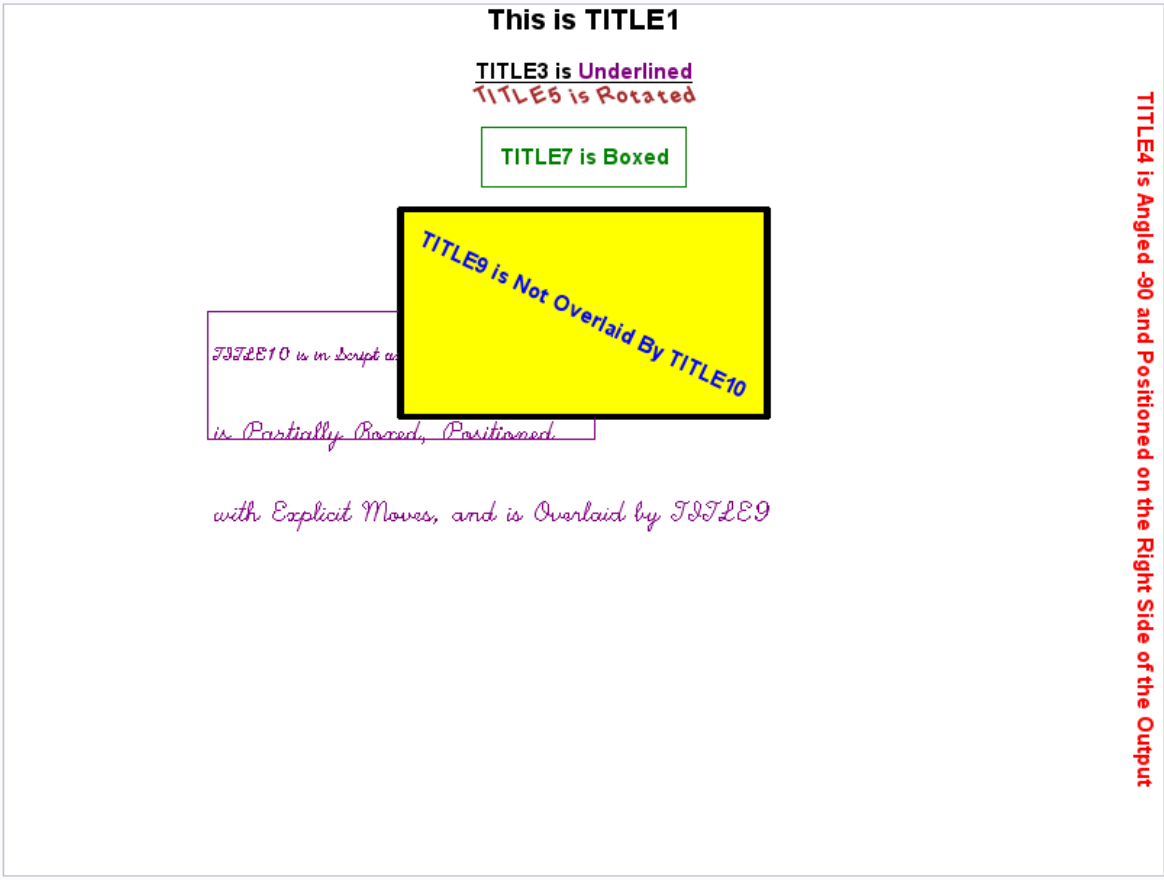
Sample library member: GTIENTI1

Details

The code for this example is in SAS Sample Library member GTIENTI1. The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

This example illustrates some ways that you can format title text. The same options can be used to format footnotes.

Output



Program

```
goptions reset=all border;

title1 "This is TITLE1"
height=4;

title3 underlin=1
       "TITLE3 is"
       color=purple
       " Underlined";

title4 color=red
       angle=-90
       "TITLE4 is Angled -90 and Positioned on the Right Side of the Output";

title5
       color=brown
```

```
       rotate=25
       "TITLE5 is Rotated";

title7 color=green
       box=1
       "TITLE7 is Boxed";

title9 color=black
       box=3
       blank=yes
       bcolor=yellow
       color=blue
       angle=-25
       "TITLE9 is Not Overlaid By TITLE10";

title10 color=purple
        draw=(20,35 20,27 58,27 58,35 20,35)cells
        move=(20,32)
        font=script
        "TITLE10 is in Script and "
        move=(20,27)
        height=2
        "is Partially Boxed, Positioned"
        move=(20,22)
        height=2
        "with Explicit Moves, and is Overlaid by TITLE9";

proc gslide;
run;
quit;
```

Program Description

Set the graphics environment. BORDER draws a border around the graph.

```
goptions reset=all border;
```

Define TITLE1. TITLE1 uses the default font and height defined in the default style. The HEIGHT= option sets the height of the text.

```
title1 "This is TITLE1"
height=4;
```

Define TITLE3. The UNDERLIN= option underlines both text strings.

```
title3 underlin=1
       "TITLE3 is"
       color=purple
       " Underlined";
```

Define TITLE4. The ANGLE= option tilts the line of text clockwise 90 degrees and places it at the right edge of the output, centered vertically.

```
title4 color=red
       angle=-90
       "TITLE4 is Angled -90 and Positioned on the Right Side of the Output";
```

Define TITLE5. The ROTATE= option rotates each character in the text string at the specified angle.

```
title5
       color=brown
       rotate=25
       "TITLE5 is Rotated";
```

Define TITLE7. The BOX= option draws a green box around the text with the thinnest of the 4 available box lines.

```
title7 color=green
       box=1
       "TITLE7 is Boxed";
```

Define TITLE9. The BLANK= option prevents the boxed title from being overwritten by TITLE10. The first COLOR= option specifies the color of the box border, and the BCOLOR= option specifies the background color of the box. The second COLOR= option specifies the text color.

```
title9 color=black
       box=3
       blank=yes
       bcolor=yellow
       color=blue
       angle=-25
       "TITLE9 is Not Overlaid By TITLE10";
```

Define TITLE10. In this statement, the DRAW= option draws a box around the first two text strings. The BOX= option is turned off by the MOVE= option that uses absolute coordinates and causes a text break.

```
title10 color=purple
        draw=(20,35 20,27 58,27 58,35 20,35)cells
        move=(20,32)
        font=script
        "TITLE10 is in Script and "
        move=(20,27)
        height=2
        "is Partially Boxed, Positioned"
        move=(20,22)
        height=2
        "with Explicit Moves, and is Overlaid by TITLE9";
```

Display titles. All existing titles are automatically displayed by the procedure.

```
proc gslide;
run;
quit;
```

Example 2: Creating a Simple Web Page with the ODS HTML Statement

Features: ODS HTML statement options

BODY=

GOPTIONS statement options

BORDER
RESET=

LEGEND statement options

ACROSS=
LABEL=

Sample library member: GONCSWB1

Details

The code for this example is in SAS Sample Library member GONCSWB1. The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

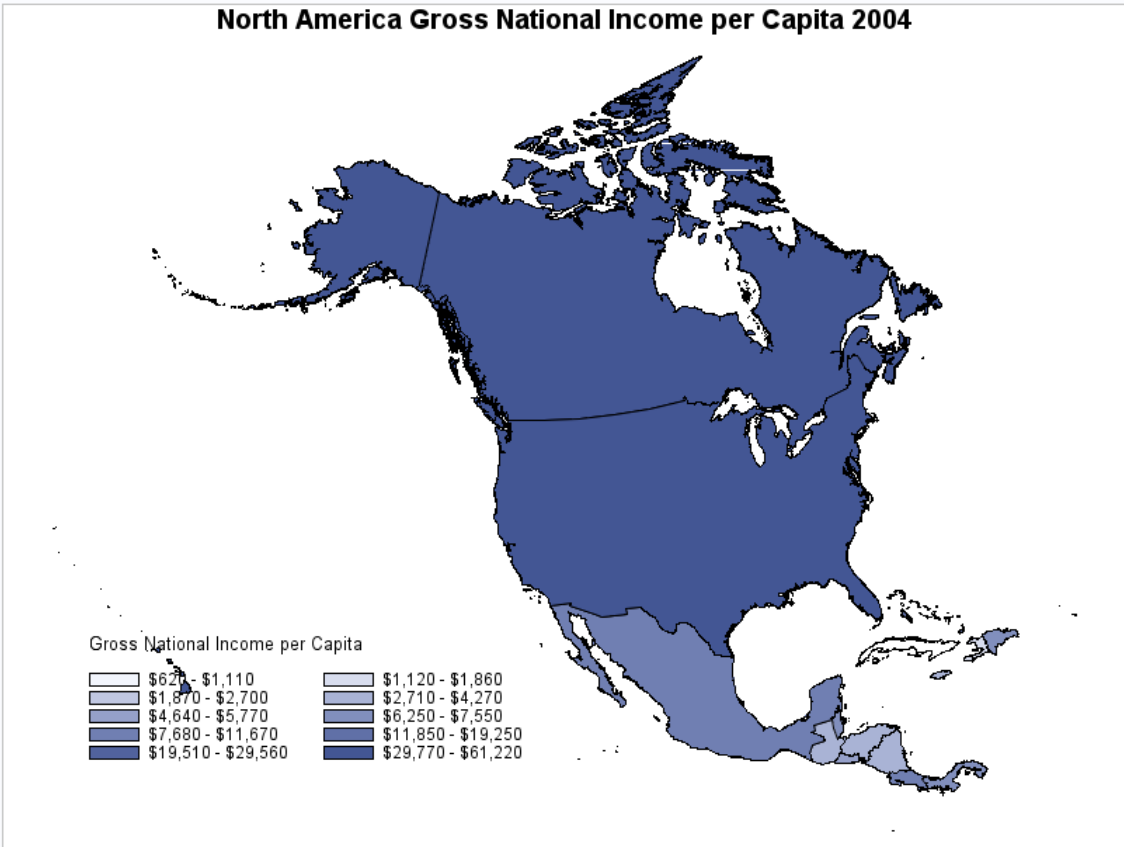
This example illustrates the simplest way to use the ODS HTML statement to create an HTML file and a PNG file that you can display in a web browser. This example also illustrates:

- the default pattern behavior with maps and explicit placement of the legend on the graph
- how the default solid map pattern uses different shades of the default style color to differentiate between countries
- how to use a LEGEND statement to arrange and position a legend so that it fits well with the graph's layout.

This example generates one body file that displays one piece of SAS/GRAPH output a map of average per capita income.

Output

Output 3.1 *Displaying a Map in a Web Page*



Program

```
goptions reset=all border;

ods html body="na_body.html";

title1 "North America Gross National Income per Capita 2004";

legend across=2
       origin=(8,5)
       mode=share
       label=(position=top
             justify=left
             "Gross National Income per Capita")
;

proc gmap map=maps.namerica data=sashelp.demographics;
      id cont id;
          format gni dollar10.0;
      choro gni / levels=10 legend=legend1;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Set the ODS HTML destination. The BODY= option names the file for storing HTML output.

```
ods html body="na_body.html";
```

Define title for the map. By default, any defined title is included in the graphics output (PNG file).

```
title1 "North America Gross National Income per Capita 2004";
```

Define legend characteristics. The ACROSS= option defines the number of columns in the legend. The ORIGIN= option specifies on the procedure output area the position of the lower left corner of the legend box. The MODE=SHARE option displays the legend in the procedure output area along with the map. The LABEL= option specifies a legend label and left-justifies it above the legend values.

```
legend across=2
       origin=(8,5)
       mode=share
       label=(position=top
             justify=left
             "Gross National Income per Capita")
;
```

Generate the prism map. Because the NAME= option is omitted, SAS/GRAPH assigns the default name GMAP to the GRSEG entry in the graphics catalog. This is the name that is assigned to the PNG file created by the ODS HTML statement.

```
proc gmap map=maps.namerica data=sashelp.demographics;
      id cont id;
          format gni dollar10.0;
      choro gni / levels=10 legend=legend1;
run;
quit;
```

Chapter 4

SAS System Options Used by SAS/GRAPH Mapping

Introduction to System Options

This chapter provides a detailed description of the system options used with SAS/GRAPH mapping software. The descriptions provide the syntax, defaults, and related options for each option.

The system options are listed alphabetically.

System options are instructions that affect the processing of an entire SAS program or interactive SAS session from the time the option is specified until it is changed. The three SAS system options included here specify the location of a particular SAS library that contains SAS/GRAPH map data sets.

For detailed information about using SAS System Options, see *SAS System Options: Reference*

Dictionary

MAPS= System Option

Specifies the location of the SAS library that contains SAS/GRAPH map data sets.

Valid in: Configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Graphics: Driver Settings

PROC OPTIONS GROUP= GRAPHICS

Note: This option can be restricted by a site administrator. For more information, see "Restricted Options" in *SAS System Options: Reference*.

Tip: You can use the APPEND or INSERT system options to add additional *location-of-maps*.

See: "MAPS System Option: UNIX" in *SAS Companion for UNIX Environments*

"MAPS System Option: Windows" in *SAS Companion for Windows*

Syntax

MAPS=*location-of-maps*

Syntax Description

location-of-maps

specifies either a physical path, an environment variable, or a libref to locate the SAS/GRAPH map data sets.

Default MAPSSAS

See Also

- "Using SAS/GRAPH Map Data Sets" on page 269

System Options:

- "APPEND= System Option" in *SAS System Options: Reference*
- "INSERT= System Option" in *SAS System Options: Reference*

MAPSGFK= System Option

Specifies the location of the SAS library that contains SAS/GRAPH digital vector map data sets. The libref MAPSGFK is assigned using the option value specified by the MAPSGFK= system option.

Valid in: Configuration file and SAS invocation

Category: Graphics: Driver Settings

PROC OPTIONS GROUP= GRAPHICS

Restriction: This option cannot be saved to the SAS registry by using the OPTSAVE procedure. Nor can it be saved by using the DMOPTSAVE command in the SAS windowing environment.

Note: This option can be restricted by a site administrator. For more information, see "Restricted Options" in *SAS System Options: Reference*.

Syntax

MAPSGFK=*location-of-maps*

Syntax Description

On the command line or in a configuration file, the syntax is specific to your operating environment. For more information, see the SAS documentation for your operating environment.

location-of-maps

specifies either a physical path, an environment variable, or the MAPSGFK libref to locate the SAS/GRAPH digital vector map data sets. The option value can consist of up to 2048 characters. Upper or lower case characters are accepted as entered. Though not required, if you specify start and end quotation marks or parentheses, these are accepted and retained as part of the option's value.

Default None

Notes The configuration file at installation time sets the MAPSGFK= option value to a physical name (path). The physical name of MAPSGFK= should not be reassigned. However, to use the MAPSGFK digital vector map data with your existing programs, issue either the "APPEND= System Option" in *SAS System Options: Reference* or the "INSERT= System Option" in *SAS System Options: Reference* system option in conjunction with the MAPS= system option. The APPEND= system option adds a new value to the end of the current value of the MAPS= system option. The INSERT= system option adds a new value as the first value of the MAPS= system option.

If you specify an environment variable for the option value, it is not expanded.

See Also

System Options

- "MAPS= System Option" on page 81
- "MAPSSAS= System Option" on page 83

MAPSSAS= System Option

Specifies the location of the SAS library that contains SAS/GRAPH non-digital map data sets. The libref MAPSSAS is assigned using the option value specified by the MAPSSAS system option.

Valid in: Configuration file and SAS invocation

Category: Graphics: Driver Settings

PROC OPTIONS GROUP= GRAPHICS

Restriction: This option cannot be saved to the SAS registry by using the OPTSAVE procedure. Nor can it be saved by using the DMOPTSAVE command in the SAS windowing environment.

Note: This option can be restricted by a site administrator. For more information, see "Restricted Options" in *SAS System Options: Reference*.

Syntax

MAPSSAS=*location-of-maps*

Syntax Description

On the command line or in a configuration file, the syntax is specific to your operating environment. For more information, see the SAS documentation for your operating environment.

location-of-maps

specifies either a physical path, an environment variable, or the MAPSSAS libref to locate the SAS non-digital vector map data sets to use with SAS/GRAPH. The option value can consist of up to 2048 characters. Upper or lower case characters are accepted as entered. Though not required, if you specify start and end quotation marks or parentheses, these are accepted and retained as part of the option's value.

Default None

Note The configuration file at installation time sets the MAPSSAS= and MAPS= option values to the same physical name (path). The MAPSSAS= option value should not be reassigned.

See Also

System Options

- "MAPS= System Option" on page 81
- "MAPSGFK= System Option" on page 82

Part 3

SAS/GRAPH Mapping Procedures

Chapter 5

GEOCODE Procedure

Overview: GEOCODE Procedure

About the GEOCODE Procedure

Geocoding is the process of adding geographic coordinates (latitude and longitude values) to an address. This process provides a way to convert address data into map locations. The geographic coordinates typically represent the center of a ZIP code, a city, an address, or any geographic region. After geocoding, the coordinates can be used to display a point on a map or to calculate distances. Geocoding also enables you to add attributes values such as census blocks to the geocoded address.

The GEOCODE procedure processes geographic information for the following entities:

- street addresses
- cities
- U.S. ZIP codes and ZIP+4 extension codes
- foreign postal codes

- custom variables in the data set, such as sales territories
- Internet Protocol (IP) addresses or other ranges

  Note: The process of adding coordinates for IP addresses is usually called geolocating. IP data is a form of range data and was not designed to be geographic. For more information, see "Understanding Range Geocoding " on page 122.

About the Required Input Data

The GEOCODE procedure requires two types of SAS data sets:

input address data sets
: contain variables that relate to specific geographic locations. For example, a data set might contain mailing address variables such as ZIP codes and street addresses, or custom geographic variables such as sales regions.

lookup data sets
: contain reference variables and geographic coordinates. For example, a lookup data set for the ZIP method contains ZIP codes and the geographic coordinates that are associated with the ZIP codes. Some geocoding methods require multiple lookup data sets. This data is essential to transform address data into location information that can be viewed on a map.

  Lookup data sets can also contain attribute variables containing data about the locations.

  For each observation in the input data set, the GEOCODE procedure attempts to match the address variable value to a value in the lookup data set. To increase the chances of a match, default lookup variable values in lookup data sets supplied by SAS are always normalized; converted to uppercase with special characters and spaces removed. These data sets are either available with the SAS release or available for download from the SAS Maps Online web site.

  If you choose to geocode with any non-default lookup variable that is contained in a lookup data set supplied by SAS, there is a significantly decreased chance of matching data.

  The GEOCODE procedure normalizes the default lookup variable values in lookup data sets that are not supplied by SAS. The address information in your input data set is also automatically normalized by PROC GEOCODE.

  The GEOCODE procedure is not shipped with all the lookup data that you might require. In some cases, you must download or purchase the data. You can download lookup data sets from the SAS Maps Online website. For more information, see "SAS Maps Online Website" on page 90.

SAS provides macro code programs to import some third-party data. Example macros are %GEOBASE2GEOCODE, %TIGER2GEOCODE, %ABS2GEOCODE, and %CODEPOINT2GEOCODE. The macros and their accompanying documentation are available for download from the SAS Maps Online web site. For street lookup data, make sure that you download the version of %TIGER2GEOCODE that corresponds to your current version of SAS. The two versions of the %TIGER2GEOCODE macro create street lookup data sets that differ in format depending on the release of SAS with which they are associated. Street lookup data sets created with the latest SAS version of %TIGER2GEOCODE cannot be used by earlier releases of PROC GEOCODE. For more information, see "SAS Maps Online Website" on page 90.

See Also

- "%GCDMEL9 Autocall Macro" on page 120
- "%MAXMIND Autocall Macro" on page 124
- %GEOBASE2GEOCODE macro program for importing Canadian roadway and address data is described in "Obtaining Street Lookup Data Sets" on page 100
- %TIGER2GEOCODE macro program for importing TIGER shapefiles for specific states and counties is described in "Obtaining Street Lookup Data Sets" on page 100
- %ABS2GEOCODE and %CODEPOINT2GEOCODE macro programs for importing postcode data from Australia and Great Britain are described in "Non-U.S. Postcodes" on page 118
- "Deciding Which Lookup Data to Use" on page 91

SAS Maps Online Website

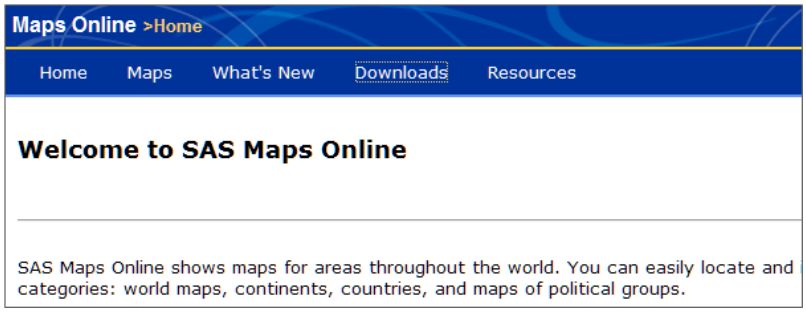
The SAS Maps Online web site contains map-related information for areas throughout the world. You can easily locate and identify specific regions in each of the following categories: world maps, continents, countries, and maps of political groups.

The website contains the following:

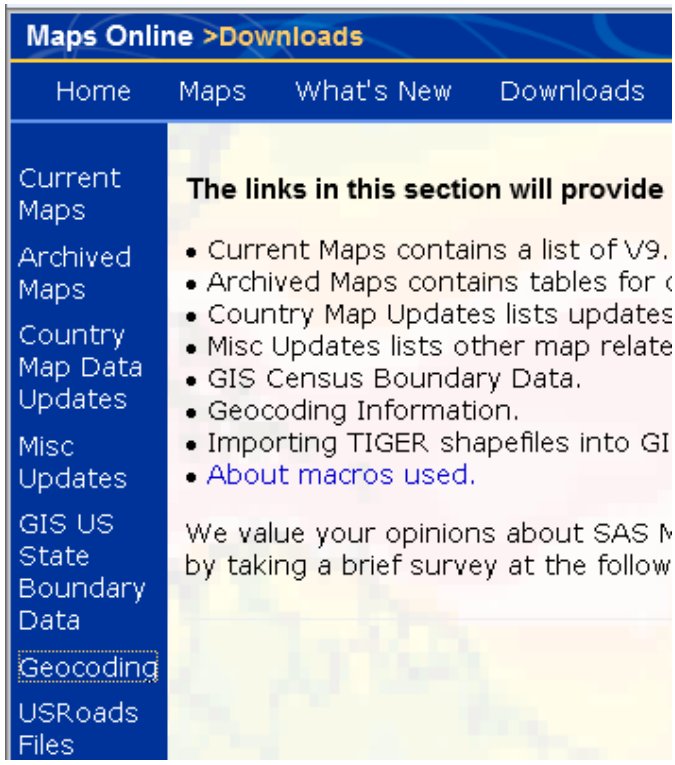
- archived maps from previous releases
- sample programs
- recent mapping and geocoding updates
- geocoding examples, techniques, and lookup data
- macro programs for importing PROC GEOCODE lookup data

The SAS Maps Online website can be accessed at http://support.sas.com/rnd/datavisualization/mapsonline/index.html. Click on the world image on the page to enter **Maps Online**.

Click the **Downloads** link in the banner to obtain the geocoding downloads mentioned in this chapter. (To download SAS macro code programs and other tools, click the **Resources** link in the banner and then click **Tools** in the left navigation bar.)



Click the **Geocoding** link in the left navigation bar on the Downloads page to access the geocoding downloads.



You can click on any of the available links in the left navigation bar to access other information, such as archived maps or recent mapping and geocoding updates.

About the Output Data

The GEOCODE procedure adds matching geographic coordinates to the observations in the output data set. In addition, the GEOCODE procedure adds a variable named _MATCHED_ that indicates how the coordinates were found. When STREET geocoding, additional variables are also output to denote the matching status for each address observation. You can also choose to add variables from the lookup data set to the output data set by using the ATTRIBUTEVAR= option.

For more information, see "Understanding Output Data" on page 92.

Deciding Which Lookup Data to Use

The type of geocoding you want to do determines the type of lookup data that is required. For example, if you want to geocode street data, you can choose between U.S. or Canadian street geocoding.

Granularity of information is an important consideration in determining which geocoding process to use. For example, does the location need to be an actual house location, or is a ZIP code or even a city center sufficient? If you are viewing the addresses on a state or U.S. map, then the ZIP code or city location is probably accurate enough.

The age of the lookup data also affects your decision. How current does the data need to be? Street address data frequently changes with the addition of new roads and changes to postal codes. The older your lookup data, the more likely it is that some address matches might be incorrect or missed completely. On the other hand, city and state lookup data rarely change.

The more up-to-date, accurate, and fine-grained the data, the more it costs to purchase and maintain. Also, higher-resolution data requires more disk storage space and takes longer to geocode. There are free sources for some types of data, but these might not be updated as frequently as the data you purchase.

It is important to remember that both purchased and free lookup data might give incorrect results. There are no guarantees with any geocoding lookup data, so the results should be carefully reviewed.

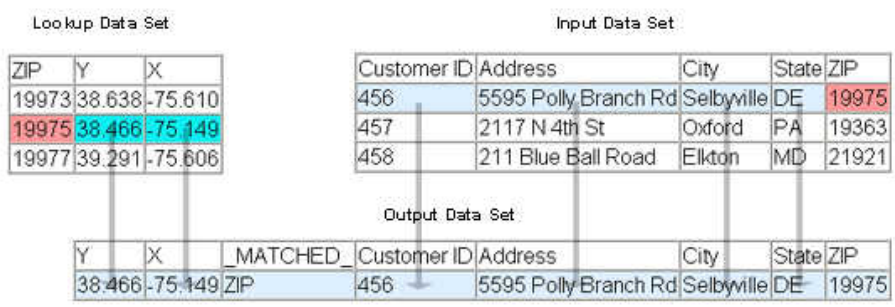
Understanding Output Data

How Geocoded Data Is Produced

When the GEOCODE procedure finds a match in the lookup data set, the procedure adds the associated coordinates to the observation in the output data set. Depending on the lookup data set, geocoded location variables can be Y and X or latitude and longitude.

The following image shows how the GEOCODE procedure obtains coordinates for the output data set by matching the ZIP code in the input data set:

Figure 5.1 *Geocoding with ZIP Codes*



The GEOCODE procedure also adds a variable named _MATCHED_ that indicates how the coordinates were found. Possible values for the _MATCHED_ variable are as follows:

State mismatch
: This value is used during international CITY geocoding. The geocoder can return multiple matches when searching for city and country. You can specify an optional state variable to help determine which of the multiple matches is the correct one. This value indicates that the geocoder cannot resolve multiple matches in different states.

Street
: A match was found for either the street address and ZIP code or the street address, city, and state.

ZIP
: A match was found for the ZIP code.

ZIP+4
: A match was found for the ZIP code and ZIP+4 extension.

ZIP mean
: Multiple observations in the lookup data set specified with the PLUS4 geocoding method matched the five-digit ZIP code and the matching latitude and longitude coordinate values were averaged.

City
: A match was found for the city and state.

City mean
: Multiple observations in the SASHELP.ZIPCODE or user-supplied lookup data set matched the city and state. In either case, the matching latitude and longitude coordinate values were averaged.

variable-name
: For CUSTOM and RANGE geocoding, a variable name indicates that a match was found for that variable.

None
: No match was found for the address.

For each observation in the input data set, the GEOCODE procedure attempts to match the address variable value to a value in the lookup data set. For most geocoding methods, the lookup data set is expected to contain only one matching observation. For example, the SASHELP.ZIPCODE lookup data set contains only one observation for each ZIP code. If the lookup data set contains multiple matches, then results can vary. The geocoding method used determines whether the first matching observation is returned or further processing is done.

Some geocoding runs might make multiple match attempts on an address. For example, if you are using ZIP code geocoding and no match is found, then the GEOCODE procedure attempts to find a matching city and state pair. Sometimes a ZIP code is not found in the SASHELP.ZIPCODE lookup data set. In this case the GEOCODE procedure searches for a matching city and state pair in the appropriate CITY lookup data set. When the GEOCODE procedure uses either the STREET or PLUS4 geocoding method and no match is found, it then searches for a five-digit ZIP code match.

Output Data Sets

By default, the GEOCODE procedure produces an output data set that contains all of the variables from the input address data set. The data set also contains the X, Y, or LONG, LAT, and _MATCHED_ variables. The X or LONG and Y or LAT coordinates must be in the same coordinate system as the lookup data set. The lookup data coordinate system is typically based on world latitudes and longitudes, but coordinate values in a specific map projection can also be used. If you want to use a different coordinate system for the output, you can convert the geocoded coordinates using a projection system application such as the GPROJECT procedure.

The default name for the output data set is DATA*n*, where *n* is the smallest integer that makes the name unique. For example, if the DATA1 data set already exists, then the default name for the output data set is DATA2. To override the default name, you can specify the output data set name with the option "OUT=*output-data-set*" on page 144.

The label of the output data set contains the text, "Geocoded *date*" where *date* is the date on which the output was created. This text is appended to the label from the input data set, if one exists.

For the STREET geocoding method, additional variables are included in the output data set. See "Output Variables for Street Geocoding" on page 100.

Adding Variables to the Output Data Set

You can specify that non-geocoding variables from the lookup data set be added to the output data set by using the ATTRIBUTEVAR= option in the PROC GEOCODE statement. For example, if you are using SASHELP.ZIPCODE as the lookup data set, then you could assign the county name (COUNTYNM) to each matched observation in the output data set.

See Also

"ATTRIBUTEVAR=(*variable-1*, *variable-2*, …)" on page 136

Understanding Street Geocoding

Overview of Street Geocoding

The street geocoding method computes geographical coordinates for specified U.S. or Canadian street addresses. This method converts a full street address that includes a house or building number, street name, city, state (or province), and ZIP (or postal) code to a map location. This method requires additional lookup data sets and additional options.

The GEOCODE procedure initially attempts to match the street name and ZIP or postal code. If no match is found, then the GEOCODE procedure attempts to match the street name, city name, and two-character postal abbreviation for state. If the second match fails, then the ZIP and the CITY geocoding methods are tried in succession. You can disable this cascading behavior by using the NOZIP and NOCITY options in the GEOCODE statement.

If a street match is found, X or LONG and Y or LAT coordinate values are interpolated along the street by using the house number.

About Street Input Data

The street input data set should minimally contain the street address and ZIP or postal code for each observation. In addition to U.S. streets, Canadian street data can be geocoded. Data that also includes the city and state or province provides a fallback way of finding a street level match if the initial attempt fails. The city, and state or province data must be presented in character format rather than a FIPS code.

For best results, use the most complete street addresses possible in your input data set. For example, "111 North Main Street" is more likely to find a match than "111 Main Street" or "111 North Main."

About Street Lookup Data

Overview of the Required Data Sets

The format of the lookup data sets varies from that of previous releases. In support of the Canadian street geocoding, the FIPS codes in the lookup data sets are replaced with city and state or province names in character format. The geocoder for the current release

cannot use lookup data sets from prior releases containing FIPS codes. Some of these lookup data sets are installed with SAS/GRAPH and others must be downloaded or created. SAS provides macro programs to import certain types of data and create these other lookup data sets. See "Obtaining Street Lookup Data Sets" on page 100 for more information.

The STREET geocoding method requires several lookup data sets:

- the 'M' or "street matching data set" on page 95
- the 'S' or "street segment data set " on page 96
- the 'P' or "street coordinate data set " on page 96
- the "street state data set" on page 97
- the "street type data set " on page 98
- the "street directional data set" on page 99

street matching data set

contains street names, either ZIP codes or postal codes, state or province names, and references to observation numbers for the street segment data set.

The STREET geocoding method searches this street matching data set for the current street name and the ZIP or postal code, or city, or state values. The FIRST and LAST variables are then used to read the observations in the street segment data set that apply to this street.

The FIRST variable identifies the starting observation in the street segment data set. The LAST variable identifies the ending observation in the street segment data set that is associated with the street match.

Default	the SASHELP.USM data set
Restriction	The name of this data set can be any valid SAS name, but it must end with the letter M.
Notes	The default data set is not installed with SAS/GRAPH. You can download the data set from the SAS Maps Online web site.
	The example SASHELP.GEOEXM data set is installed with SAS. This data set is a street matching data set for Wake County, North Carolina (NC), USA.
	When creating a customized version of the data, set you must use the same variable names, data types, order, and simple indexes that are used in the default data set.
	SAS provides macro programs to import additional data and write them to street matching data sets.
Tips	If you move the location of the street lookup data set, you must also move the associated index file. If this index file is not present, a warning is issued in the SAS log by PROC GEOCODE.
	Any non-default street matching data set is specified by the LOOKUPSTREET= option. When you specify this option for the M data set, the library must also contain two corresponding data sets whose names end with S (segment) and P (coordinate) respectively.
See	"LOOKUPSTREET=*street-matching-data-set* " on page 140

For more information about these macros, and the default, installed, and created data sets, see "Obtaining Street Lookup Data Sets" on page 100.

street segment data set

contains variables to identify the street type suffix and prefix, and street direction prefix and suffix.

Each street segment is associated with a range of house numbers along one side of the street. The range is specified by the SIDE, FROMADD, and TOADD variables. The START variable identifies the first observation in the street coordinate data set that is associated with the street segment. The N variable specifies the number of observations in the street coordinate data set containing the coordinates that delineate the street segment.

The street segment data sets that are provided by SAS contain attribute variables with additional information pertaining to the street segment and side. U.S. Census tracts and county names in character format are examples of such attribute variables.

You can specify that non-geocoding variables from the street segment lookup data set be added to the output data set by using the ATTRIBUTEVAR= option in the PROC GEOCODE statement.

Default the SASHELP.USS data set

Restriction The name of this data set must be the same as the street matching data set name, except the last character must be S instead of M.

Notes The default data set is not installed with SAS/GRAPH. You can download the data set from the SAS Maps Online web site.

The SASHELP.GEOEXS data set is installed with SAS. This data set is a street segment data set for Wake County, North Carolina (NC), USA.

When creating a customized version of the data set, you must use the same variable names and data types that are used in the default data set.

SAS provides macro programs to import the street type prefixes and write them to this street segment data set.

See For more information about writing out non-geocoding variables from the lookup data set, see "Adding Variables to the Output Data Set" on page 94.

"ATTRIBUTEVAR=(*variable-1*, *variable-2*, …)" on page 136

For more information about these macros, and the default, installed, and created data sets, see "Obtaining Street Lookup Data Sets" on page 100.

street coordinate data set

contains latitude and longitude coordinates.

Default the SASHELP.USP data set

Restriction The name of this data set must be the same as the street matching data set name, except the last character must be P instead of M.

Notes The default data set is not installed with SAS/GRAPH. You can download the data set from the SAS Maps Online web site.

The SASHELP.GEOEXP data set is installed with SAS. This data set is a street coordinate data set for Wake County, North Carolina (NC), USA.

SAS provides macro programs to create this street coordinate data set with specific data.

See For more information about these macros, and the installed, alternate, and created data sets, see "Obtaining Street Lookup Data Sets" on page 100.

street state data set

contains state character names rather than FIPS numeric codes. The character names can be any of the following:

- Postal service two-character code, such as 'NC'
- non-abbreviated state name, such as 'North Carolina'
- nonstandard state ID, such as 'N. Car.' or 'No. Car.'

This data set, as delivered with SAS, contains common state alias values for state IDs or Canadian provinces that are not postal service abbreviations. However, other nonstandard state IDs can be included in a customized copy of the data set, or in the input address data set.

The STREET or CITY geocoding methods can use this street state data set. In some instances a match cannot be found by using the street name and ZIP or postal code. In this case, the STREET geocoding method uses the street state data set to determine the complete state or province name. It also uses the state name or ID provided in the input address. For example, this data set enables the geocoder to equate the state postal abbreviation of TX, or the nonstandard abbreviation of Tex., with the complete state name of Texas. In another example a non-abbreviated state name might be specified in the input address data set. In this case the CITY geocoding method uses the street state data set to find a matching two-character state ID. This state ID is a necessary component of its matching algorithm.

Default SASHELP.GCSTATE. This data set contains all of the United States and its overseas territories, and all Canadian provinces.

Notes When creating a customized version of this data set you must use the same variable names, data types, and indexes that are used in the SASHELP.GCSTATE data set. It is no longer necessary to maintain the same sort order of the variables from the original data set.

The optional StateAlias variable contains nonstandard values of state or province abbreviations; those not used by the postal service. In one observation, multiple values delimited by a vertical bar '|' can be specified. For example, nonstandard designations for the state of California are ' CF | Cal. | Cali. | Calif.'.

When you create a customized version of this data set you can use duplicate values of the MapIDNameAbrv variable. These values are state or province postal service abbreviations. These values provide a key to the geocoder when it groups equivalent state or province names. However, do not use the same abbreviation for unrelated states to avoid misleading

matches. For example, do not use the abbreviation 'MI' to represent both the states of Mississippi and Michigan.

The values of the variables MAPIDNAME (state or province name), and MAPIDNAME2 (normalized state or province name) are changed. They match, respectively, the ISONAME and ISONAME2 variables found in the MAPSGFK.WORLD_CITIES data set.

Tips If you move the location of the street state lookup data set, you must also move its associated SASHELP.GCSTATE.INDEX data set. If this data set index is not present, a warning is issued in the SAS log by PROC GEOCODE.

Any non-default street state data set is specified by the STATE= option.

See "STATE=*data-set*" on page 146

For more information about these macros, and the installed and the created data sets, see "Obtaining Street Lookup Data Sets" on page 100.

street type data set

contains street type suffixes and prefixes. The STREET geocoding method uses the street type data set to convert street type suffixes and prefixes from the input address observation to standardized forms.

Default the SASHELP.GCTYPE data set, which is installed with SAS/GRAPH

Notes The Canadian version of this data set, GCTYPE_CAN, contains standard street type prefixes most commonly used in Canadian addresses. Examples include Bannockburn Road and Boulevard Quebec. The GROUP variable equates multiple abbreviations for the same street type. For example, AVENUE can be abbreviated as 'AV' in provinces using French and as 'AVE' in provinces using English. The data set is created by the %GEOBASE2GEOCODE import program when Canadian lookup data is imported.

You might want to create a customized version of the data set. When doing so, you must use the same variable names and data types that are used in either the SASHELP.GCTYPE data set or the GCTYPE_CAN data set. An index is not provided or required for the SASHELP.GCTYPE data set. To create a data set for another nation, the data set name must be GCTYPE_*xxx*, where *xxx* is the three-character country abbreviation. A customized version of the data set must contain types consisting of more than one character. For example, to get more street matches in a location where nonstandard street type abbreviations are common, you could add them to a custom version of GCTYPE. Examples might include 'WK' for Walkway, 'PW' for Parkway, or 'AL' for Alley. For an example of how to create a custom GCTYPE data set, see "Example 3: Street Geocoding" on page 156. However, adding ambiguous abbreviations to GCTYPE can create confusion when parsing addresses. For example, 'BL' is already in GCTYPE as an abbreviation for 'BOULEVARD'. Therefore, you should not add an instance of 'BL' as an abbreviation for 'BLUFF'.

The SASHELP.GCTYPE data set contains standard U.S. Postal Service street type suffixes, such as Avenue and Drive. In this data set, the GROUP variable equates multiple abbreviations for the same street type. For example, AVENUE can be abbreviated as 'AVN' or 'AVE'.

SASHELP.GCTYPE includes some suffixes that are not in the USPS table but are found in U.S. Census Bureau TIGER/Line data for various U.S. localities. If addresses in your geocoding input data contain unusual or nonstandard suffixes, then create a custom version of this data set that includes those suffixes. No simple index is required for the GCTYPE or GCTYPE_CAN data set. The complete list of USPS street types can be found in Appendix C of the USPS Publication 28, Postal Addressing Standards.

Tip Any non-default street type data set is specified by the TYPE= option. For example, use this option to specify the GCTYPE_CAN data set that contains the Canadian street prefixes and suffixes when geocoding Canadian street addresses.

See "TYPE=*data-set*" on page 146

For more information about these macros, and the installed, alternate, and created data sets, see "Obtaining Street Lookup Data Sets" on page 100.

street directional data set

contains various U.S. or Canadian street direction text strings used as prefixes and suffixes in street names. Examples in the U.S. data set are east, southeast, SE, and north. An example in the Canadian version of this data set is nord. Nord is the French directional string meaning north. This data set is able to be edited.

Default the SASHELP.GCDIRECT data set, which is installed with SAS/GRAPH. This data set contains U.S. street direction text strings.

Notes The Canadian version of this data set, GCDIRECT_CAN, contains English and French direction names and abbreviations for Canadian streets as used by the Canada Post. An example is Nord-Ouest, the French direction name for northwest. GCDIRECT_CAN is created by the %GEOBASE2GEOCODE import program.

You might want to create a customized version of the data set. When doing so you must use the same variable names, data types, order, and simple index that are used in either the SASHELP.GCDIRECT data set or the GCDIRECT_CAN data set. To create a data set for another nation, the data set name must be SASHELP.GCDIRECT_*xxx*, where *xxx* is the three-character country abbreviation.

SAS provides the %GEOBASE2GEOCODE macro program to create or modify this street directional data set with specific data.

Tips If you move the location of the street directional lookup data sets, you must also move their associated data set index– for example, SASHELP.GCDIRECT.INDEX. If this data set index is not present, a warning is issued in the SAS log by PROC GEOCODE.

Any non-default street direction data set is specified by the DIRECT= option. For example, use this option to specify the GCDIRECT_CAN data set that contains Canadian directional strings.

See "DIRECTION=*data-set*" on page 136

For more information about these macros, and the installed, alternate, and created data sets, see "Obtaining Street Lookup Data Sets" on page 100.

Obtaining Street Lookup Data Sets

PROC GEOCODE in the second maintenance release for SAS 9.4 and later cannot read the earlier versions of the lookup data sets. Likewise, PROC GEOCODE in releases prior to the second maintenance release for SAS 9.4 cannot read the newer lookup data sets. The street lookup data sets are reformatted. Most notably, to enable Canadian street geocoding, the StateFp and PlaceFp variables containing FIPS codes are replaced with state and city character names. With the creation of lookup data sets that contain state variables with full names, variables such as MapIDName are no longer needed. The STREET, and CITY geocoder can process state variables with alias names that are not used by the postal service. SASHELP.GCSTATE and SASHELP.GCDIRECT are examples of the required lookup data sets used by the GEOCODE procedure's STREET matching method. To avoid a warning message in the SAS log, ensure that you obtain the lookup data sets that correspond to the version of SAS that you are running.

The default street matching data sets (USM, USS, and USP) are not installed with SAS/GRAPH. These data sets contain address lookup data for the entire United States. You can download these data sets from the SAS Maps Online website. For more information, see "SAS Maps Online Website" on page 90.

The USM, USS, and USP data sets are created from U.S. Census Bureau TIGER/Line shapefiles. After the annual release of new TIGER/Line data, updated versions of the USM, USS, and USP data sets can be downloaded from the SAS Maps Online web site.

The GEOEXM, GEOEXS, and GEOEXP data sets in the SASHELP library are installed with SAS/GRAPH by default. These data sets contain street method lookup data for Wake County, North Carolina (NC), USA. They can be used for trial geocoding tests to determine whether to download and install the nation-wide lookup data sets from the SAS Maps Online web site.

SAS offers macro programs to import lookup data. Two of these macro programs create the lookup data sets used by the GEOCODE procedure's STREET method of geocoding. The %GEOBASE2GEOCODE macro program imports Canadian roadway and address data from their GeoBase data clearinghouse. The macro program also creates the Canadian versions of the street type and direction data sets. The %TIGER2GEOCODE macro program imports TIGER/Line shapefiles from the U.S. Census Bureau for specific states and counties. This macro program creates the USM, USS, and USP data sets, as well as the U.S. street type and direction data sets. Download the appropriate version of these macro programs from the SAS Maps Online web site to create or modify the lookup data sets needed for street method geocoding. The version that you download depends on whether you are geocoding with SAS 9.4 or later, or a release prior to SAS 9.4.

The data sets created with the macro programs corresponding to SAS 9.4 or later are in a different format than those of prior SAS releases. For example, street lookup data sets in prior releases contained FIPS codes while the current data sets do not.

For more information see "SAS Maps Online Website" on page 90.

Output Variables for Street Geocoding

In addition to the default output variables, the STREET geocoding method creates the following variables in the output data set:

M_ADDR
: contains the street address for a street match. The M_ADDR value is the match value from the lookup data.

MATCHED
: indicates how the coordinates were found. The following values are used with the _MATCHED_ variable:

State mismatch
: This value is used during international CITY geocoding. The geocoder can return multiple matches when searching for city and country. You can specify an optional state variable to help determine which of the multiple matches is the correct one. This value indicates that the geocoder cannot resolve multiple matches in different states.

Street
: A match was found for either the street address and ZIP code or the street address, city, and state.

ZIP
: A match was found for the ZIP code. The GEOCODE procedure uses the ZIP method when a ZIP+4 method finds no match between ZIP+4 values and when a single ZIP code match is found

ZIP+4
: A match was found for the ZIP code and ZIP+4 extension. The location is the center of the street segment.

ZIP mean
: Multiple observations in the lookup data set specified with the PLUS4 geocoding method matched the five-digit ZIP code and the matching latitude and longitude coordinate values were averaged. The GEOCODE procedure uses this method when no matches are located with the ZIP+4 values.

City
: A match was found for the city and state. The GEOCODE procedure uses this method when either a specified ZIP or ZIP+4 method fails to return a match

City mean
: Multiple observations in the SASHELP.ZIPCODE lookup data set matched the city and state and the matching latitude and longitude coordinate values were averaged.

variable-name
: For CUSTOM and RANGE geocoding, a variable name indicates that a match was found for that variable.

None
: No match was found for the address.

M_CITY
: contains the city name for a city and state match. The M_CITY value is the match value from the lookup data.

M_STATE
: contains the two-character postal abbreviation for a city and state match. The M_STATE value is the match value from the lookup data.

M_ZIP
: contains the ZIP code value for a ZIP level match from the lookup data.

M_OBS
: contains the row number for the matching observation from the primary lookup data set.

STATUS
indicates the type of match that was found. The following values are used with the _STATUS_ variable:

City and State Match
The street address did not match but a match was found for the city name and two-character postal abbreviation.

Found
The street address matched.

ZIP Match
The street address did not match but a match was found for the ZIP code.

(Blank)
No match was found.

NOTES
contains tokens that provide additional information about the match. For more information, see "Street Geocoding Note Values" on page 102.

SCORE
Contains a numeric value indicating the relative accuracy of the match.

Street Geocoding Note Values

The STREET geocoding method creates a _NOTES_ variable in the output data set. This variable provides details about the quality of the address match by using token strings. For example, the value "AD ZC NM" contains three tokens that indicate that the street name, ZIP code, and house number matched.

Each token in the _NOTES_ value has an associated value, and the sum of these values makes up the value of the _SCORE_ variable.

The following table displays the tokens and their scores:

Table 5.1 *Tokens for the _NOTES_ Variable*

Token	Score	Description
AD	20	The street name matched.
ADSDP	5	Address Direction Suffix matched lookup data Direction Prefix.
ADPDS	5	Address Direction Prefix matched lookup data Direction Suffix.
CT	5	The city name matched.
DP	15	The street direction prefix matched.
DS	15	The street direction suffix matched.
ENDNM	0	The house number was outside the ranges of values in the lookup data set for the matching street. The geocoded coordinates for the nearest end of the street were used.

Token	Score	Description
MCS	0	Multiple matches were found for the input street address and the street, city, and state in the street segment lookup data set.
MVP	0	The street geocoder detected missing values for the X or Y coordinates in the user-supplied lookup data set.
MZC	0	Multiple matches were found for the street address and ZIP code.
NM	10	The house number matched on the correct side of the street.
NMOS	5	The house number matched an address range in the lookup data set, but is on the opposite side of the street from the matched range.
NOADD	0	An invalid street address was input.
NOCTM	0	A match was found using the input address' street and ZIP code. However, it was noted that the input city value was different from the city on the matched lookup data set observation. For example, the input data set's address contained an incorrect city, "100 Main St., Raleigh, NC 27513". However, the matching lookup data set observation had the correct city value, "100 Main St., Cary, NC 27513".
NODPA	-10	The input address had no direction prefix but the matching street did have a direction prefix. For example, the input street name was "Main St." but the matching street was "N Main St."
NODPM	-15	The input address had a direction prefix but it either did not match the direction prefix of the matching street or the matching street had no direction prefix. For example, the input street name was "North Main St." but the matching street was "Main St."
NODSA	-10	The input address had no direction suffix but the matching street did have a direction suffix. For example, the input street name was "Johnson Ave" but the matching street was "Johnson Ave S."

Token	Score	Description
NODSM	-15	The input address had a direction suffix but it either did not match the direction suffix of the matching street or the matching street had no direction prefix. For example, the input street name was "Johnson Ave South" but the matching street was "Johnson Ave."
NOLNM	0	The lookup data set contains missing values for the house numbers of the matching street. The geocoded coordinates for the center of the matching street were used.
NONM	0	The input address has no house number. The geocoded coordinates for the center of the matching street were used.
NOSTM	0	A match was found using the input address' street and ZIP code. However, it was noted that the input state value was different from the state on the matched lookup data set observation. For example, the input data set's address contained an incorrect state, "100 Main St., Cary, ND 27513". However, the matching lookup data set observation had the correct state value, "100 Main St., Cary, NC 27513".
NOTPA	-10	The input address had no street type prefix, but the matching address did have a street type prefix. For example, the input address was "110 Quebec." but the matching address was "110 Boulevard Quebec"
NOTPM	-20	The street type prefix of the input address was either not the same as the type prefix of the matching street or the matching street had no type prefix. For example, the input street name was "Boulevard Quebec" but the matching street name was "Avenue Quebec".
NOTSA	-10	The input address had no street type suffix, but the matching address did have a street type suffix. For example, the input address was "110 Main." but the matching address was "110 Main St."
NOTSM	-20	The street type suffix of the input address was either not the same as the type suffix of the matching street or the matching street had no suffix. For example, the input street name was "Park St." but the matching street name was "Park Ave."
NS	0	The input address had no state value.

Token	Score	Description
NSM	0	An initial match was found for the input city and country pair in the CITY lookup data set. However, the state variable value of the matching observation was not a match.
NOZC	0	No ZIP code was provided.
NOZCM	0	A match was found using the input address' street, city, and state. However, it was noted that the input ZIP code value was different from the ZIP on the matched lookup data set observation. For example, the input data set's address contained a transposed ZIP code, "100 Main St., Cary, NC 25713". However, the matching lookup data set had the correct ZIP code value, "100 Main St., Cary, NC 27513".
ST	5	The two-character state abbreviation matched.
TP	20	The street type prefix matched.
TS	20	The street type suffix matched.
ZC	15	The five-digit ZIP code or non-U.S. postal code matched.

Tips for Street Geocoding

The following table contains suggestions and comments for the STREET geocoding method.

Category	Suggestions and Comments
Most recent software and lookup data	Install the most recent SAS release. The STREET method is being continuously upgraded and each release contains improved matching results. Installing the most recent SAS release also updates the existing data sets such as SASHELP.GCTYPE. They are used in STREET geocoding. The formats of the STREET lookup data sets have changed from that of releases prior to SAS 9.4. Download the lookup data sets that are not installed with SAS/GRAPH from the SAS Maps Online website. They are differentiated by version. You can use only the lookup data sets that correspond to your SAS release.
	Obtain the most recent street lookup data sets. The nationwide U.S. lookup data available on the SAS Maps Online web site is updated annually when the U.S. Census Bureau releases new TIGER/Line shapefiles. These include the USM, USS, and USP data sets. To support U.S. and Canadian street geocoding, state and direction data sets are available, as well as Canadian street type and direction data sets. The %TIGER2GEOCODE macro program imports TIGER/Line shapefiles from the U.S. Census Bureau. This program provides more street matches for any specific SAS release. The %GEOBASE2GEOCODE macro program imports Canadian roadway and address data from the GeoBase data site. Both macro programs create the lookup data sets used by the GEOCODE procedure's STREET method. Download these macro programs from the SAS Maps Online web site. For more information see "SAS Maps Online Website" on page 90.

Category	Suggestions and Comments
Correct data	Examine the input address values for observations that do not get a street match. Here are some common reasons why street matches are not found: • The address is a P.O. box or contains apartment information. (See the text immediately below this bulleted list for details.) • The direction suffix (or prefix) is positioned such that is it not detected. For example, the input address might be ‘Green Level West Road ’ instead of ‘Green Level Road West.’ • The street name is not in the TIGER or GeoBase data that is being used. • The city or street name in the lookup data uses an alternate spelling. An example is Hillsboro versus Hillsborough. • Data is misspelled for street or city names. • Digits are incorrect or transposed for house numbers or ZIP or postal codes. The city name on the target street in the lookup data is within the city limits of one city but the mailing address uses a different city. Including the adequate amount of data can avoid this problem. The address processing code attempts to strip apartment numbers, suite numbers, floor numbers, mailbox numbers, P.O. box numbers, and other non-street related address elements. However, depending on how these elements are interspersed in the address string, the GEOCODE procedure might not remove them. You might need to remove them yourself. *Note:* Do not remove house numbers.
Adequate amount of data	Include as many elements of a mailing address as possible. The street name is required. Include the house number, the ZIP code, the city, and the state values whenever possible.

Category	Suggestions and Comments
ZIP or postal codes	Always include the ZIP or postal code with the input address, if available. Street geocoding is most effective when using the ZIP or postal code. If ZIP is omitted, the city and state values must be used, and they can be misleading. If the ZIP or postal code is included, but incorrect, the street geocoder uses city and state values along with a street name. A ZIP or postal code might be incorrect for locations in the data set where the U.S. Postal Service has reassigned local ZIP codes when modifying its delivery routes. For example, when a ZIP match is not found, an input address with its direction prefix at the end of the street name, '101 Harrison Ave N', is matched to '101 N Harrison Ave' in the lookup data. Conversely, when a ZIP match is not found, an input address with its direction prefix at the beginning of the street name, '101 N Harrison Ave', is matched to '101 Harrison Ave N' in the lookup data. *Note:* When street geocoding with Canadian data that does not contain postal codes, disable the geocoder cascading to a secondary ZIP matching method by specifying the NOZIP option. The city value of a street segment from the original TIGER data is not the mailing address city. Instead, it denotes whether that segment is physically within the corporate limits of a city. A missing city value means that the street segment is not within any incorporated city. For example, a house with a Leon, Iowa mailing address is in an unincorporated portion of Decatur County, Iowa. The TIGER data for this house might have a missing city value despite the house having a valid mailing address. Also, it is common for a house to be inside one city's limits but its mail is delivered from another city's post office branch. For example, there are streets near SAS that are within Apex, North Carolina, but have a Cary, North Carolina mailing address. Specifying the ZIP code for all input addresses can prevent geocoding problems that result from this situation.
Missing lookup data set	If you change the location of the lookup data sets, always move the data set index that correlates to them. A warning is issued in the SAS log by the GEOCODE procedure if an index file is expected and not present.
	The STREET method locates an address using the street name and ZIP or postal code or by using the street name with the city and state or province. A ZIP or postal code represents less data than a city. As a result, when you use these codes the geocode procedure runs faster and provides more reliable locations.

Understanding City Geocoding

Overview of City Geocoding

This section describes details about geocoding with either U.S. or international city names. The requirements for input data and lookup data vary depending on which area you choose. SAS provides a default lookup data set for each type of geocoding, but an alternate data set can be specified instead.

About City Input Data Sets

For U.S. city geocoding each observation in your input data must contain a city and a state. The recommended state value is the two-character state abbreviation but the complete state name can also be used.

For worldwide geocoding each observation in your input address data must include the city name and the country. The country value can be the complete country name or its two- or three-character country abbreviation. An example is GB or GBR for the United Kingdom. An optional state, province, or region name can also be specified.

About U.S. City Lookup Data

The default lookup data set for U.S. CITY geocoding is MAPSGFK.USCITY_ALL. This data set is licensed by SAS from GfK GeoMarketing GmBH, who is the single source of this data and all of its updates. The data set is covered by the GfK GeoMarketing copyright, is provided with SAS/GRAPH in the MAPSGFK library, and is updated for each SAS release.

You can download updates of the MAPSGFK.USCITY_ALL data set from the SAS Maps Online website. For more information, see "SAS Maps Online Website" on page 90.

You can specify that non-geocoding variables from the lookup data set be added to the output data set by using the ATTRIBUTEVAR= option in the PROC GEOCODE statement.

The default lookup data set MAPSGFK.USCITY_ALL contains the following variables:

Default Name:
    Label:

STATE
    Numeric State code

COUNTY
    Numeric County code

CITY
    City name

ID
    Region code

ID1
    Admin1 code

ID2
Admin2 code

X
Projected longitude using the experimental Miller II projection method

Y
Projected latitude using the experimental Miller II projection method

CITY2
Normalized CITY name for geocoding

Note: Values are converted to uppercase and all spaces and characters that are not alphabetic or numeric are stripped

STATECODE
State postal code

COUNTY_NAME
County Name

LAT
Unprojected degrees latitude (WGS 84)

LONG
Unprojected degrees longitude (WGS 84)

Capital
Capital city

POP_TYPE
GfK pop2010 category

FEATURE_ID
Census Government ID

ALT_M
Altitude in meters

ALT_FT
Altitude in feet

About World City Lookup Data

The default lookup data set for world CITY geocoding is MAPSGFK.WORLD_CITIES. This data set contains more than 200,000 cities worldwide. The data set is licensed by SAS from GfK GeoMarketing GmBH, who is the single source of this data and all of its updates. The data set is covered by the GfK GeoMarketing copyright, is provided with SAS/GRAPH in the MAPSGFK library, and is updated for each SAS release.

You can download updates of the MAPSGFK.WORLD_CITIES data set from the SAS Maps Online web site. An unabridged data set named MAPSGFK.WORLD_CITIES_ALL, with more than one million observations, is also available for downloading. The unabridged data set is updated annually. For more information, see "SAS Maps Online Website" on page 90.

You can specify that non-geocoding variables from the lookup data set be added to the output data set by using the ATTRIBUTEVAR= option in the PROC GEOCODE statement.

The default lookup data set MAPSGFK.WORLD_CITIES and the unabridged data set MAPSGFK.WORLD_CITIES_ALL both contain the following variables:

Name:
Label:

X
Projected longitude using the experimental Miller II projection method.

Y
Projected latitude using the experimental Miller II projection method.

ID
Country Alpha code.

MapIDName2
Normalized state or province name for geocoding.

Note: Values are converted to uppercase and stripped of all spaces and characters that are not alphabetic or numeric.

CITY2
Normalized CITY name for geocoding.

Note: Values are converted to uppercase and stripped of all spaces and characters that are not alphabetic or numeric.

CONT
Numeric number for continent.

ISONAME
ISO country name.

CITY
City name.

CtType
POP categories for cities where applicable.

Rank
Grouping of CtType high to low.

Vintage
Recorded Year of data.

LONG
Unprojected degrees longitude (WGS 84).

LAT
Unprojected degrees latitude (WGS 84).

ISO
ISO country code.

ISOALPHA2
ISO Alpha2-code for country.

ISOALPHA32
ISO Alpha3-code for country.

MapID
ID from MAP data set.

Note: The map data set specified in the SAS/GRAPH GMAP procedure

MapLevel
Map level.

IDNAME
Country name.

MapIDName1
ID1NAME from MAP data set

About Alternate U.S. City Lookup Data

The GEOCODE procedure uses the MAPSGFK.USCITY_ALL as the default lookup data set for U.S. CITY geocoding. However, you can use the LOOKUPCITY= option to specify an alternate lookup data set. For example, if you want to exclude Puerto Rico and the U.S. Virgin Islands, use the MAPSGFK.USCITY data set. The data set is covered by the GfK GeoMarketing copyright and is provided with SAS/GRAPH in the MAPSGFK library. The data set is updated for each SAS release. Another alternate data set is SASHELP.ZIPCODE, which is made available with SAS/GRAPH in the SASHELP library. See "About the SASHELP.ZIPCODE Lookup Data Set" on page 115 for details. SASHELP.ZIPCODE was the original lookup data set used in CITY geocoding prior to SAS 9.4.

The lookup data set SASHELP.ZIPCODE contains multiple observations for cities. When using this data set, the city location is found by averaging the ZIP code data for the city and state to find the mean location.

Alternate lookup data sets can be used automatically by the geocoder to help find equivalent state names if the specified non-abbreviated state name does not match. Such an alternate lookup data set is SASHELP.GCSTATE. It contains city and state or province names in character format. The geocoder looks in this data set if it needs additional state data in order to make a match between a non-abbreviated state name and a state code.

Any other alternate lookup data set for U.S. CITY geocoding must contain the following information. These exact variable names are not required, but the variable values must contain the following information:

CITY
Name of the city (does not have to be normalized)

Note: Values are not converted to uppercase and spaces and non-alphabetic and non-numeric characters remain intact.

STATECODE
Two-character abbreviation of the state name

Note: Instead of a two-character STATECODE, you have the option of using the complete state name. However, the state values in the alternate lookup data set must match the values that are in your input address data. The state names do not have to be normalized, meaning that values are not converted to uppercase and spaces and non-alphabetic and non-numeric characters remain intact. When using a custom lookup data set, the GEOCODE procedure converts to uppercase the state name and removes special characters and spaces.

LAT
Latitude of the city center

LONG
Longitude of the city center

About Alternate World City Lookup Data

The GEOCODE procedure uses the MAPSGFK.WORLD_CITIES as the default lookup data set for world CITY geocoding. However, you can use the LOOKUPCITY= option

to specify an alternate lookup data set. An example alternate data set is MAPSGFK.WORLD_CITIES_ALL, which is made available for download from the SAS Maps Online web site.

Any other alternate lookup data set for worldwide CITY geocoding must contain the following information. These exact variable names are not required, but the variable values must contain the following information:

CITY
: Name of the city

ISOALPHA2 or ISOALPHA3
: Two- or three-character abbreviation of the country name

Note: Instead of a two- or three-character country abbreviation, you can use the complete country name. However, the country values in the alternate lookup data set must match the values that are in your input address data.

LAT
: Latitude of the city center

LONG
: Longitude of the city center

The alternate world CITY geocoding lookup data set can also include an optional state, province, or region name. This allows the GEOCODE procedure to differentiate between cities that share the same name but are located in different regions. The state, province, or region names in the lookup data do not have to be normalized, meaning converted to uppercase and stripped of spaces and non-alphabet and non-numeric characters.

Tips for City Geocoding

The following table contains suggestions and comments for using the CITY geocoding method.

Category	Suggestions and Comments
Most recent software and lookup data	Installing the most recent SAS release updates the following data sets that can be used by the GEOCODE procedure: • MAPSGFK.USCITY_ALL • MAPSGFK.WORLD_CITIES • SASHELP.ZIPCODE MAPSGFK.WORLD_CITIES_ALL is an unabridged version of world cities available for download. See "SAS Maps Online Website" on page 90 for details. You can also download quarterly updates of SASHELP.ZIPCODE from the SAS Maps Online website. See "SAS Maps Online Website" on page 90 for details.

Category	Suggestions and Comments
Correct data	Here are some common reasons why CITY matches are not found: • The city name is misspelled. • The city name in the lookup data uses an alternate spelling (for example, Hillsboro versus Hillsborough). • The two-character state ID is transposed (for example, YN instead of NY). • The non-U.S. city name in the lookup data uses a local spelling. MAPSGFK.WORLD_CITIES is an example of a lookup data set with local spellings. • There are multiple cities with that name in the country. Use the ADDRESSSTATEVAR= option to specify a state, province, or region variable to differentiate the cities. • The town can be very small and not included in the MAPSGFK.WORLD_CITIES data. Download the more complete MAPSGFK.WORLD_CITIES_ALL data set. See "SAS Maps Online Website" on page 90 for details. • The state (or province) variable is missing from the input address data set and you are geocoding within one country. The state variable is required in order for the geocoder to differentiate states or provinces sharing cities with the same name. • The state (or province) variable is missing from the input address data set. Or the customized state lookup data set when a state or province is specified with the ADDRESSSTATEVAR= option. When international city geocoding, use of the state variable is optional. However, when a state variable is specified, the geocoder tries to match the specified state with states supplied in the input data set(s).

Understanding ZIP Code Geocoding

Overview of ZIP Code Geocoding

The terms ZIP code and postal code are used interchangeably. Both terms refer to a group of linear mail delivery routes. ZIP code is the United States Postal Service's designation while postal code is used by other national post offices. ZIP codes can also be assigned to a single building or to a post office. There are also U.S.P.S. military post office designations that associate a ZIP code with either a branch of the military or with a U.S. embassy. Whereas a ZIP code is not considered a polygonal area, a county, state, or country is considered such.

Generally, ZIP code address data specifies a center location for the ZIP code. The center approximates the geographic center of the ZIP code. The Postal Service does not create official ZIP code boundaries. Each data vendor draws its own boundaries, which causes ZIP center variations among different vendors' products.

The basic ZIP code in the United States is five digits. ZIP code data provides the general vicinity of an address, but usually not the actual street. If you want to locate the street,

you can try using ZIP+4 geocoding. For more information, see "Understanding ZIP+4 Geocoding " on page 119.

About ZIP Code Input Data

Your input data must contain a valid ZIP or postal code for each observation. If a ZIP code is not found, then the GEOCODE procedure attempts to find the city center location. If you are interested in the ZIP code location only, you can turn off this cascading behavior using the NOCITY option in the GEOCODE statement.

About ZIP Code Lookup Data

ZIP code geocoding uses the SASHELP.ZIPCODE lookup data set by default. You can use the LOOKUP= option to specify alternate data sources. SASHELP.ZIPCODE can also be used as an alternate lookup data set for the CITY method. SASHELP.ZIPMIL is an alternate lookup data set that contains ZIP codes designating either a U.S. military branch or embassy. You can also import postal codes outside the U.S. from other sources of data. To import postal codes, you can use the IMPORT procedure or a DATA step, depending on the format of the third-party data.

After you import non-U.S. postal codes with centroids into a SAS data set, you can use this data set as the lookup data set with the ZIP method. For an explanation of the sources of free Australian and British postal code locations, see "Non-U.S. Postcodes" on page 118.

When providing the geocoder with your own lookup data set of ZIP or postal codes, you must set any longitude and latitude variables containing missing values to a period (.). Do not set these missing values to zero (0), because these are valid values that indicate a location at the equator. SAS prepares the SASHELP.ZIPMIL data set in such a manner. You can use this data set for validating APO, FPO, and DPO values or you can create an alternate file by merging it with the ZIP code data set.

You can specify that non-geocoding variables from the lookup data set be added to the output data set by using the ATTRIBUTEVAR= option in the PROC GEOCODE statement. In the case of military ZIP codes, you can assign the STATE name value using the ATTRIBUTEVAR= option to each matched observation in the output data set. This would indicate the location of the ZIP code if there are no latitude and longitude values.

About the SASHELP.ZIPCODE Lookup Data Set

The default lookup data set for ZIP code geocoding is SASHELP.ZIPCODE. This data set can also be used as an alternate lookup data set with the CITY method for U.S. locations. SASHELP.ZIPCODE is provided with Base SAS and is updated for each SAS release.

You can download quarterly updates of the SASHELP.ZIPCODE data set from the SAS Maps Online website. For more information, see "SAS Maps Online Website" on page 90.

SASHELP.ZIPCODE contains the following variables:

Name:
Label:

ZIP
The five-digit ZIP code.

Y
Latitude (decimal degrees) of the center of the ZIP code.

X
Longitude (decimal degrees) of the center of the ZIP code.

ZIP_CLASS
ZIP code Classification: P=PO Box, U=Unique zip used for large organizations or businesses or buildings, Blank=Standard or non-unique.

CITY
Name of the city or organization.

STATE
Two-digit number (FIPS code) for the state or territory.

STATECODE
Two-letter abbreviation for the state name.

STATENAME
Full name of the state or territory.

COUNTY
FIPS county code.

COUNTYNM
Name of county or parish.

MSA
Metropolitan Service Area code by common population pre-2003; no MSA for rural areas.

AREACODE
Single Area Code for the ZIP code.

AREACODES
Multiple Area Codes for the ZIP code.

TIMEZONE
Time Zone for the ZIP code.

GMTOFFSET
Difference (hours) between GMT and Time Zone for the ZIP code.

DST
ZIP code observes Eastern Time. Y is Yes and N is No.

PONAME
USPS Post Office Name: same as City.

ALIAS_CITY
USPS – Alternate names of city separated by "||".

ALIAS_CITYN
Local – alternate names of city separated by "||".

CITY2
Clean CITY name for geocoding

Note: Values are converted to uppercase and all spaces and characters that are not alphabetic or numeric are stripped.

STATENAME2
Clean STATENAME for geocoding

Note: Values are converted to uppercase and all spaces and characters that are not alphabetic or numeric are stripped.

About Alternate ZIP Code Lookup Data

The SASHELP.ZIPCODE data set is the default lookup data set for the GEOCODE procedure. SAS also supplies SASHELP.ZIPMIL as an alternate lookup data set. However, data from other sources can be used as long as it is read into a SAS data set. You can use the "LOOKUP=*lookup-data-set*" on page 137 option to specify alternate data sources.

The SASHELP.ZIPMIL alternate lookup data set contains the same variables as the SASHELP.ZIPCODE data set. However, the CITY variable contains the military acronyms AFO, FPO, and DPO, and the ALIAS_CITY variable contains those acronym descriptors. The longitude and latitude variables contain missing values, and as such cannot provide a geocoded location for a military ZIP code. However, you can programmatically identify these as military ZIP codes as you geocode. For example, use the "ATTRIBUTEVAR=(*variable-1*, *variable-2*, …)" on page 136 option of the PROC GEOCODE statement to assign the STATE value to your military ZIP code.

- Army Post Office (APO) address information and ZIP codes associated with Army or Air Force installations.
- Fleet Post Office (FPO) address information and ZIP codes associated with Navy installations or ships.
- Diplomatic Post Office (DPO) address information and ZIP codes associated with U.S. embassies overseas.

For ZIP code geocoding, any lookup data set must contain the following values but the exact variable names are not required:

Default Name:
Description:

ZIP
Five-digit ZIP code

X
Longitude of the center coordinate

Y
Latitude of the center coordinate

CITY2
City name values must be converted to uppercase and all spaces and characters that are not alphabetic or numeric are stripped.

STATENAME2
State name values must be converted to uppercase and all spaces and characters that are not alphabetic or numeric are stripped.

Note: The character values in your input and lookup data sets do not need to be a case-sensitive match. Character value matching in the GEOCODE procedure is not case sensitive.

Note: The geocoder converts alphabetic characters to uppercase and strips all spaces and special characters from the input data set. Performing this same cleanup in your lookup data set helps the geocoder find more matches during its comparison of the lookup and input data sets.

Additional attribute variables can also be in the alternate lookup data set. You can specify that non-geocoding variables from the lookup data set be added to the output data set by using the ATTRIBUTEVAR= option in the PROC GEOCODE statement.

Non-U.S. Postcodes

Postcode data from other countries can be used with the ZIP method if it includes either longitude and latitude or X and Y coordinates. You have to import postcode data into a SAS data set, which becomes the lookup data set for the ZIP method. If the postcodes contain alpha characters, the characters should be converted to uppercase. Remove any spaces and punctuation from the postcodes.

In addition, make sure that the geographic system of the imported data is compatible with your needs. For example, if you want geocoded locations in the World Geographic System 1984 (WGS 84) geodetic datum, your imported values should be in that datum. If they are not, then you must apply either the appropriate coordinate conversions or datum transformations or both.

Great Britain's national mapping agency, the Ordnance Survey (OS), provides location data for 1.7 million Royal Mail postcodes in their free Code-Point Open product. The SAS macro program %CODEPOINT2GEOCODE imports the OS files. The GEOCODE procedure uses these OS files to locate British mailing addresses by postcode. The macro also converts the British National Grid coordinates to World Geodetic System 1984 (WGS 84) longitude and latitude.

The Australian Bureau of Statistics (ABS) produces a generalized map of Australian postal areas. The SAS macro program %ABS2GEOCODE imports those postcode areas, computes the centroids, and creates a lookup data set for use by the GEOCODE procedure. The macro also creates a SAS/GRAPH map data set of the postal area polygons.

The %CODEPOINT2GEOCODE and %ABS2GEOCODE macro programs and their accompanying documentation are available for download from the SAS Maps Online website. For more information, see "SAS Maps Online Website" on page 90.

U.S. Military ZIP Codes

ZIP codes for U.S. military post offices are provided in the ZIPMIL data set in the SASHELP library. You can combine this data set with the ZIPCODE data set to support military ZIP codes.

Tips for ZIP Code Geocoding

The following table contains suggestions and comments for the ZIP geocoding method.

Category	Suggestions and Comments
Most recent software and lookup data	Installing the most recent SAS release updates the SASHELP.ZIPCODE data set that is used in ZIP geocoding. You can also download quarterly updates of SASHELP.ZIPCODE from SAS Maps Online. See "SAS Maps Online Website" on page 90.
Correct data	Remove all data entry errors from your addresses, if possible. For example, transposed digits in the input ZIP code result in no match, or worse, return a completely wrong location.

Category	Suggestions and Comments
Non-U.S. postal codes	Remove any spaces and punctuation and uppercase any characters.

Understanding ZIP+4 Geocoding

Overview of ZIP+4 Geocoding

With ZIP+4 geocoding, the GEOCODE procedure attempts to match the five-digit ZIP code and ZIP+4 extension from your address data set with the lookup data set.

If a ZIP+4 code is not found, the GEOCODE procedure attempts to match the standard five-digit ZIP code. If that is not found, the procedure attempts to find the city. If you are interested in the ZIP code location only, you can turn off this cascading behavior using the NOCITY option in the GEOCODE statement.

About ZIP+4 Lookup Data

For ZIP+4 code geocoding, you can use data that is derived from the TIGER ZIP+4 files that are available from the SAS Maps Online website.

Note: The U.S. Census Bureau has omitted ZIP+4 values from TIGER files released after 2006. The lookup data set to be used with the PLUS4 geocode method was created using 2006 TIGER data and is available on SAS Maps Online. The lookup data set will be updated after the Census Bureau reinstates ZIP+4 values into TIGER files. For more information about the status of the ZIP+4 data, and how to download the current data, see "SAS Maps Online Website" on page 90.

You can also purchase the GEO*Data product containing current ZIP+4 centroids, available from Melissa Data at this website: http://www.melissadata.com/reference-data/geodata.htm.

SAS includes an autocall macro (%GCDMEL9), which imports Geo*Data files into SAS data sets.

You can specify that non-geocoding variables from the lookup data set be added to the output data set by using the ATTRIBUTEVAR= option in the PROC GEOCODE statement.

About Alternate ZIP+4 Lookup Data

When you use ZIP+4 geocoding, you must specify an alternative lookup data set because SASHELP.ZIPCODE does not contain any ZIP+4 values. This data set must contain the following variables:

Default Name:
Description:

ZIP
Five-digit ZIP code

PLUS4
: Four-digit ZIP+4 extension

X
: Longitude of the central coordinate

Y
: Latitude of the central coordinate

You can specify different names for the variables by using options in the PROC GEOCODE statement. For example, the LOOKUPPLUS4 option specifies the name of the ZIP+4 extension variable in the lookup data set.

The ZIP and PLUS4 variables can contain either character data or numeric data. The lookup data type must match the type of the corresponding variable in your input data set.

Note: The character values in your input and lookup data sets do not need to be a case-sensitive match. Character value matching in the GEOCODE procedure is not case sensitive.

Additional non-geocoding attribute variables can also be in the alternate lookup data set. You can add these variables to the output data set by using the ATTRIBUTEVAR= option in the PROC GEOCODE statement.

You can obtain a lookup data set for ZIP+4 geocoding from the SAS Maps Online website at **`http://support.sas.com/rnd/datavisualization/mapsonline/index.html`**. On the Downloads page, select **Geocoding** to access the downloads that are related to geocoding.

An alternative source for ZIP+4 lookup data is the Geo*Data product from Melissa Data. You can use the %GCDMEL9 autocall macro to convert Geo*Data files to SAS data sets. For more information, see "%GCDMEL9 Autocall Macro" on page 120.

%GCDMEL9 Autocall Macro

Overview of the %GCDMEL9 Autocall Macro

The %GCDMEL9 autocall macro enables you to directly import Geo*Data files from Melissa Data as SAS data sets. Geo*Data files contain third-party ZIP+4 lookup data for use with PLUS4 geocoding.

Geo*Data files are available for each state. The files are provided as text files within compressed (ZIP) archives. Melissa Data also provides the PKUNZIP utility to extract the text files.

The %GCDMEL9 macro uses the following macro variables:

DATASETNAME
: specifies the name of the output data set.

DATASETPATH
: specifies the location where the output data set is created.

DATASETLABEL
: (optional) specifies a label for the output data set.

LIBNAME
: specifies the name for a new library that is assigned for the location that you specified in the DATASETPATH macro variable.

UNZIPPEDPATH
: specifies the location of the extracted Geo*Data files that you want to import. The %GCDMEL9 macro attempts to read all of the text (.txt) files in this directory.

WORKPATH (Optional)
: specifies the path where temporary files are written. The default path is the path for the WORK library.

Usage Example for the %GCDMEL9 Autocall Macro

In this example, a Geo*Data file for the state of Delaware (DE.txt) was extracted to **`C:\Mydata`**. The lookup data set is created in an existing directory **`C:\Geocode`** and assigned the libref ZIP4. The resulting data set is named ZIP4.DELAWARE.

The following code imports the data:

```
/* Define macro variables */
%let UNZIPPEDPATH=C:\Mydata;
%let DATASETPATH=C:\Geocode;
%let DATASETNAME=Delaware;
%let LIBNAME=ZIP4;
%let DATASETLABEL=ZIP+4 lookup data for Delaware;
/* Submit autocall macro */
%GCDMEL9
```

Tips for ZIP+4 Geocoding

The following table contains suggestions and comments for the PLUS4 geocoding method.

Category	Suggestions and Comments
Most recent lookup data	Check the SAS Maps Online website to see whether there is an update available for the ZIP4 lookup data set. (It is to be made available when the U.S. Census Bureau restores ZIP+4 values to their TIGER/Line shapefiles.) See "SAS Maps Online Website" on page 90. You can check with the third-party vendor, Melissa Data, to see whether its Geo*Data product contains ZIP+4 values that are more recent than 2006. The files can be imported with the %GCDMEL9 AUTOCALL macro.
Correct data	Here are some common reasons why ZIP+4 matches are not found: • The input ZIP+4 contains transposed digits. • The ZIP+4 is new and therefore is not in the lookup data set that you are using.

Understanding Range Geocoding

Overview of Range Geocoding

Range geocoding matches individual address values to a lookup data set containing a range of values. IP address data is a form of range data. IP data was not designed to be geographic like street addresses. For this reason, the process of adding coordinates to IP addresses is usually called geolocating rather than geocoding.

With IPv4 address data, this method requires specifying a second data set with the RANGEDATA= option in addition to the LOOKUP= option. The second data set contains various ranges of IPv4 addresses. The location ID is the key variable that links the two data sets. This variable in the range data set can map to more than one location in the lookup data set. When geolocating with IPv6 address data, the range coordinates are contained in the lookup data set along with their matching locations. Therefore, there is no need to use the RANGEDATA= option. The one-to-one relationship between address range and location already exists.

There are no default variable names with this method. You must specify all data sets and variable names.

Generally, IP addresses are collected from visitors to websites and indicate the connection the visitor used. IP address lookup data contains information that matches ranges of IP addresses to particular geographic locations. A range of IP addresses usually belongs to a company or an Internet provider. The location found is not at the street or even ZIP code level, but might indicate the city, state, or country where the IP address is registered.

About Range Lookup Data

Range geocoding with IPv4 addresses requires a lookup data set and an additional range data set. Two data sets are required because a range of addresses can map to more than one location. Range geocoding with IPv6 addresses is possible and requires a single lookup data set that contains all geographic coordinates and the ranges of IPv6 addresses.

- The lookup data set contains geographic coordinates (latitude and longitude). Specify this data set with the LOOKUP= option.
- The range data set identifies the ranges (of IPv4 addresses or of other items). Specify this data set with the RANGEDATA= option.

A location ID key variable links the two data sets. Both data sets must contain this variable in order to identify locations for each IPv4 range. Internally, the proper range is found, and then the key value is used to access the lookup data set to find the latitude and longitude for that key. This key variable can map a range of addresses in the range data set to more than one location in the lookup data set.

Range geocoding with IPv6 addresses requires a single lookup data set that contains all geographic coordinates and the ranges of IPv6 addresses. No key variable is necessary. If you are range geolocating both IPv4 and IPv6 addresses, specify only the IPv4 data set name with the LOOKUP= option. The %MAXMIND autocall macro automatically creates the IPv6 data set name based on the IPv4 name, with the number 6 appended to it.

The lookup data set contains geographic coordinates (latitude and longitude). Specify this data set with the LOOKUP= option. When specifying the IPv6 lookup data set, make sure the number 6 is appended to the name. If you are range geolocating both IPv4 and IPv6 addresses, then provide separate data sets for each IP version.

The range data set identifies the ranges (of IPv4 addresses or of other items). Specify this data set with the RANGEDATA= option. The range data set is not required for IPv6 address data because the ranges are provided in the IPv6 lookup data set.

A KEY variable links the IPv4 lookup and range data sets. Both data sets must contain this variable in order to identify locations for each IPv4 range. Internally, the proper range is found, and then the key value is used to access the IPv4 lookup data set to find the latitude and longitude for that key. No key variable is necessary when using IPv6 addresses.

The lookup data set must contain the following variables:

- [IPv4 and IPv6] an X variable that contains the longitude value of the center coordinate. The default variable name is X.
- [IPv4 and IPv6] a Y variable that contains the latitude value of the center coordinate. The default variable name is Y.
- [IPv4 only] a key variable that corresponds to a key variable in the range data set.
- [IPv6 only] a variable that specifies the beginning value of a range of IPv6 addresses
- [IPv6 only] a variable that specifies the ending value of a range of IPv6 addresses

The IPv4 range data set must contain the following variables:

- a variable that specifies the beginning value of a range of IPv4 addresses
- a variable that specifies the ending value of a range of IPv4 addresses
- a key variable that corresponds to a key variable in the lookup data set

You can obtain lookup and range data from third-party vendors. One vendor is MaxMind, Inc. at https://www.maxmind.com/. You can use the %MAXMIND autocall macro to convert their legacy IPv4 comma-separated value (CSV) files from MaxMind into SAS data sets. These CSV files must be downloaded from the MaxMind website to your local directory before running the macro. The updated version of the %MAXMIND autocall macro is available. Use it to convert IPv6 as well as IPv4 CSV files from MaxMind into SAS data sets. This macro automatically runs a new subprogram designed to convert IPv6 data. The IPv4 data from MaxMind is provided in decimal format and in a fully qualified format with periods. MaxMind typically provides IPv6 data in collapsed hexadecimal with colons and decimal formats.

You can specify that non-geocoding variables from the lookup data set be added to the output data set by using the ATTRIBUTEVAR= option in the PROC GEOCODE statement.

The MaxMind IPv6 GeoLite lookup data contains only country-level latitude and longitude coordinates. There are many IPv6 address ranges for each country, and the coordinates for each and every address range is to the center of the country. For example, your lookup data set contains many U.S. IPv6 addresses. Your lookup ranges list as coordinates for each of these addresses a latitude of 38 degrees and a longitude of 97 degrees. This location is the center of the United States.

If you supply your own lookup data set, change any zero representing a missing value to a SAS missing numeric: a period (.). Any 32-bit IPv4 address must consist of only integers zero (0) through nine (9) and three periods (.). Any 128-bit IPv6 address must consist of only alphanumeric characters separated by a colon (:). IPv6 addresses might

be fully represented or collapsed. They cannot contain port numbers, URLs, or Fully Qualified Domain Names (FQDN).

%MAXMIND Autocall Macro

Overview of the %MAXMIND Autocall Macro

The original %MAXMIND autocall macro enables you to convert IPv4 geocoding data from MaxMind, Inc. into SAS data sets. Likewise, the updated version of the %MAXMIND autocall macro enables you to convert IPv6 data as well as IPv4 data. Either version of the %MAXMIND autocall macro supports MaxMind's IP data in its legacy, comma-separated value (CSV) format. Neither import macro supports MaxMind's next generation file format (GeoLite2).

The name of the SAS data set resulting from an IPv6 conversion is that of an IPv4 conversion with a number 6 appended to it.

The original %MAXMIND macro uses the following macro variables:

CSVPATH
: specifies the path where the MaxMind CSV files are located. You must extract the files from the ZIP archive before using the %MAXMIND autocall macro.

IPDATAPATH
: specifies the path where the output SAS data sets are created. You must have Write permission for this path.

CSVBLOCKSFILE
: specifies the filename for the CSV file that contains IPv4 address range values. The file that you specify must contain the startIpNum and endIpNum variables. The default filename is GeoLiteCity-Blocks.csv.

CSVLOCATIONFILE
: specifies the filename for the CSV file that contains longitude and latitude values. The default filename is GeoLiteCity-Location.csv.

CSVCOUNTRYFILE (Optional)
: specifies the name of the optional MaxMind CSV file that contains country names. The default filename is GeoIPCountryWhoIs.csv.

WORKPATH (Optional)
: specifies the path where temporary files are written. The default path is the path for the WORK library.

The %MAXMIND macro creates the CITYBLOCKS and CITYLOCATION data sets in the path that you specified for the IPDATAPATH variable. The libref IPDATA is created automatically for this path.

Compared to the macro variables required for the original %MAXMIND macro, the updated version of the macro requires an additional set of variables names. These names are identical, except for the number 6 appended to each. The following are the macro variables that cover IPv6 address data:

CSVPATH
: specifies the path where the MaxMind CSV files are located. You must extract the files from the ZIP archive before using the updated %MAXMIND autocall macro.

IPDATAPATH
: specifies the path where the output SAS data sets are created. You must have Write permission for this path.

CSVBLOCKSFILE6
: specifies the filename for the CSV file that contains IPv6 address range values. The file that you specify must contain the startIpNum and endIpNum variables. The default filename is GeoIPv6.csv.

CSVLOCATIONFILE6
: specifies the filename for the CSV file that contains longitude and latitude values. The default filename is GeoLiteCityv6.csv.

CSVCOUNTRYFILE6 (Optional)
: specifies the name of the optional MaxMind CSV file that contains country names. The default filename is GeoIPASNum2v6.csv.

WORKPATH (Optional)
: specifies the path where temporary files are written. The default path is the path for the WORK library.

The %MAXMIND macro and its sub-programs create the CITYBLOCKSFILE6 data set in the path that you specified for the IPDATAPATH variable. This data set, among other variables, contains the starting and ending IPv6 addresses and the longitude and latitude values. The libref IPDATA is created automatically for this path.

Usage Example for the Original %MAXMIND Autocall Macro

In this example, data from MaxMind was extracted to **`C:\Mydata`**. The output SAS data sets are created in the directory **`C:\Geocode`**.

The following code imports the data:

```
%let CSVPATH=C:\Mydata;
%let IPDATAPATH=C:\Geocode;
%let CSVBLOCKSFILE=GeoLiteCity-Blocks.csv;
%let CSVLOCATIONFILE=GeoLiteCity-Location.csv;
%let CSVCOUNTRYFILE=GeoIPCountryWhois.csv;
%MAXMIND
```

The imported data sets are IPDATA.CITYBLOCKS and IPDATA.CITYLOCATION.

Usage Example for the %MAXMIND Autocall Macro

In this example, data from MaxMind was extracted to **`C:\Mydata`**. The output SAS data sets are created in the directory **`C:\Geocode`**. The %MAXMIND macro requires all IPv4 and IPv6 data files to be downloaded from MaxMind and placed in the same location. If the location is not the same, then the macro fails to execute and issues an error.

The following code imports all the required data:

```
%global CSVPATH;
%let CSVPATH=C:\Mydata;
%let IPDATAPATH=C:\Geocode;
%let CSVBLOCKSFILE=GeoLiteCity-Blocks.csv;
%let CSVLOCATIONFILE=GeoLiteCity-Location.csv;
%let CSVCOUNTRYFILE=GeoIPCountryWhois.csv;
%let CSVBLOCKSFILE6=GeoIPv6.csv;
%let CSVLOCATIONFILE6=GeoLiteCityv6.csv;
%let CSVCOUNTRYFILE6=GeoIPASNum2v6.csv;
%MAXMIND
```

The imported data sets are IPDATA.CITYBLOCKS, IPDATA.CITYLOCATION, and IPDATA.CITYBLOCKS6. If you chose a name other than the default, then the IPv6 data set name has a 6 appended to your specified name.

Note: Because the processing of IPv6 data is optional, the range geocoder does not issue error messages for missing IPv6 addresses. The output data set with a missing IPv6 address has a corresponding _MATCHED_ variable value of **None**.

Note: A valid IPv6 address is marked as missing when no IPv6 lookup data set is available.

Output Variables for Range Geocoding

In addition to the default output variables, the RANGE geocoding method creates the following variable in the output data set:

MATCHED
: indicates how the coordinates were found. The following values are used with the _MATCHED_ variable:

Link-Local
: For CUSTOM and RANGE geocoding, no match was found for the dynamically configured single-network segment IP address. In RFC 3927 , the Internet Engineering Task Force has reserved the address block 169.254.1.0 through 169.254.254.255.

The link-local IPv6 address range translates to

```
FE80:0000:0000:0000:0000:0000:0000:0000 -
FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF (FE80::/10 prefix)
```

Private
: For CUSTOM and RANGE geocoding, no match was found for the private IP address. The Internet Assigned Numbers Authority (IANA) has reserved the following three blocks of the IP address space for private internets:

```
10.0.0.0        -   10.255.255.255  (10/8 prefix)

172.16.0.0      -   172.31.255.255  (172.16/12 prefix)

192.168.0.0     -   192.168.255.255 (192.168/16 prefix)
```

The private IPv6 address range translates to

```
 FC00:0000:0000:0000:0000:0000:0000:0000 -
FDFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF (FC00::/7 prefix)
```

variable-name
: For CUSTOM and RANGE geocoding, a variable name indicates that a match was found for that variable.

None
: No match was found for the address.

Tips for Range Geocoding

The following table contains suggestions and comments for the RANGE geocoding method.

Category	Suggestions and Comments
Most recent lookup data, in its original format	Obtain the most recent lookup data, in their legacy format, from the MaxMind website. This can be IPv4 or IPv6 address data, or both. Place the unzipped files in a common directory. Do not change the filenames. Import all the files with the %MAXMIND AUTOCALL macro. Open the macro and declare the macro variables for the IPv4 MaxMind files. The %MAXMIND AUTOCALL macro imports IPv4 data and the IPv6 data from MaxMind. The macro invokes the sub-macro programs to import any IPv4 and IPv6 MaxMind files that are located in the designated directory. The website is located at http://www.maxmind.com/en/home. Neither import macro supports MaxMind's next generation file format (GeoLite2). If you change the filenames of the MaxMind data, the %MAXMIND autocall macro and its sub-macro programs cannot recognize or import the file data.
Correct data	Here are some common reasons why IP addresses are not matched: • If an input IP address fails to match, the IP address might contain transposed digits. • IP addresses might not be in the right format. IPv4 addresses must be in either a dotted quad or a decimal format. IPv6 addresses are supported. IPv6 addresses must be hexadecimal in format with a colon (:) separator. They can be full addresses or collapsed. • The length of the IP address might be too long. The length of IPv4 addresses must be no longer than 15, 39 for IPv6, including Field Separators.
Missing data	An IPv6 address was detected but no IPv6 lookup data was supplied. Make sure your input lookup data includes a range of IPv6 addresses, specified with the LOOKUP= option, and appends a number 6 to the name. This data set must be separate from the IPv4 data set specified.

Understanding Custom Geocoding

Overview of Custom Geocoding

Custom geocoding is a flexible option that enables you to apply your own type of lookup data to your address data. For example, your input address data might include only telephone area codes for different customers. Other examples are internal sales territories and Metropolitan Statistical Areas (MSA).

Use the option "METHOD=geocoding-method" to specify CUSTOM when you want to geocode an unconventional, non-address type of lookup variable. You must use the option "ADDRESSVAR=variable" to identify the address variable in your address data. You must also use the option "LOOKUPVAR=variable" to identify the non-address variable in the lookup data set.

About Using Lookup Data with Custom Geocoding

Any data can be used as lookup data with the CUSTOM method of geocoding. The only requirement is that you have at least three variables. The variables must be the projected or unprojected coordinate values (X and Y or LAT and LONG), and include a key variable to look up.

You can specify that non-geocoding variables such as COUNTY, TIME ZONE, and AREA CODE from the lookup data set be added to the output data set. Do this by using the ATTRIBUTEVAR= option in the PROC GEOCODE statement.

Optimizing Performance

Overview of Enhancing Performance

Geocoding often requires very large lookup data sets, which can affect the performance of the GEOCODE procedure. You can optimize your geocoding performance by performing the following actions:

- Index your lookup data sets by using the appropriate variables.
- Load the lookup data sets into memory by using the SASFILE statement.
- Minimize running over a network or to an external disk drive.
- Avoid using cross-environment data access (CEDA) when accessing lookup data sets.
- Exclude the location of the WORK library from either the Microsoft Security Essentials or the Microsoft Forefront antivirus program.

Indexing Your Lookup Data Sets

SAS provides indexes for the lookup data sets that it provides. If you use alternative lookup data sets, then indexing your lookup data sets can improve performance. You should create an index by using the same variables that SAS supplies and that are appropriate for your geocoding method.

Indexes that are recommended but are missing can reduce the performance of the geocoder. If your missing index is associated with a data set supplied by SAS, then reinstalling that data set from the original source restores the index. You must add the index for a data set that is not supplied by SAS. Refer to Table 5.2 on page 129.

SAS provides indexing for the following data sets that are used with the GEOCODE procedure.

- The SASHELP.ZIPCODE data set and the ZIP4 data set from the SAS Maps Online website are optimized.
- Data sets converted with either the %GCDMEL9 or the %MAXMIND autocall macros.

  Note: The necessary sorting of data set variables is done automatically when using these macros. Be sure to use the version of the %MAXMIND autocall macro that

shipped with your SAS release. If you upgrade your SAS release, you should reimport the MaxMind CSV files using its version of the %MAXMIND macro.

Note: Data sets of IPv4 addresses converted by the original or the updated %MAXMIND autocall macro are sorted and indexed. Data sets of IPv6 addresses converted by the updated %MAXMIND autocall macro are sorted only.

- The STREET geocoding lookup data sets that are provided by SAS.

If the GEOCODE procedure accesses a data set supplied by SAS or created with a SAS import program, it checks for the presence of an index if it expects one. If an expected index is not found, a warning is printed in the SAS log. Note that this check is only for the existence of an index. The procedure does not determine whether the correct variables are in the index.

If you use SAS procedures to copy or move the lookup data sets, any associated indexes are preserved. However, if you use an operating system utility and do not also copy or move the index files, you need to rebuild the indexes. Copying an indexed data set with a DATA step does not apply the index to the new data set, so any index must be regenerated.

For ZIP+4 geocoding, you should create a simple index on the ZIP variable and a compound index on the ZIP and ZIP+4 variables.

For RANGE geocoding of IPv4 addresses you can create lookup data from a source other than MaxMind. If you do this, you are required to sort and index both the range and lookup data sets. But you need only index the data sets on the location ID. For RANGE geocoding of IPv6 addresses, you can create IPv6 lookup data from a source other than MaxMind. You are required to sort this lookup data, but an index is not required. For all versions of SAS, you must sort the range data set by the beginning IP address variable. For IPv6, the lookup data set contains the ranges of IP addresses as well as the coordinates and thus has no need for an indexing key variable. In the case of an IPv4 lookup data set, you must also sort it by the key variable that links the observations between the IPv4 range and lookup data sets. In addition, add a simple index to the IPv4 lookup data set using that key variable.

Indexing for the range data set is not required.

For more information, see "Understanding SAS Indexes" in *SAS Language Reference: Concepts*.

Index Names Used By Geocoding Data Sets

Here is a comprehensive list of the indexes used by geocoding data sets.

Table 5.2 *Geocoding Data Set Index Names and Variables*

Default Data Set Name/ Description*	Index Filename	Default Index Variable(s)**
Primary Lookup for U.S. Streets (Default name: SASHELP.USM)	Name2_Zip	NAME2, ZIP
	Name2_Zctra	NAME2, ZCTA
	Name2_MapIDNameAbrv_City2	NAME2, MAPIDNAMEABRV, CITY2

Default Data Set Name/ Description*	Index Filename	Default Index Variable(s)**
Primary Lookup for Canadian Streets (Default name: Canada_m)	Name2_MapIDNameAbrv_City2 Name2_MapIDName_City2	NAME2, MAPIDNAMEABRV, CITY2 NAME2, MAPIDNAME, CITY2
SASHELP.ZIPCODE	City2_StateCode ZIP City2_StateName2	CITY2, STATECODE ZIP CITY2, STATENAME2
Australian Postal Codes	POA_CODE	POA_CODE
British Postal Codes	PC	PC
MAPSGFK.US_CITY_ALL	StateCode_City2	STATECODE, CITY2
MAPSGFK.WORLD_CITIES or WORLD_CITIES_ALL (from SAS MapsOnline)	ISOalpha2_City2 ISOalpha3_City2 ISOname2_City2 ID	ISOALPHA2, CITY2 ISOALPHA3, CITY2 ISONAME2, CITY2 ID
ZIP4 (PLUS4 method)	ZIP_PLUS4 ZIP	ZIP, PLUS4 ZIP
CITYLOCATION	CITYLOCATION	LocId
SASHELP.GCDIRECT	DIRECTION	DIRECTION
SASHELP.GCSTATE	ISOalpha2_MapIDNameAbrv ISOalpha3_MapIDNameAbrv ISOalpha2_MapIDName2 ISOname2_MapIDNameAbrv ISOalpha3_MapIDName2 ISOname2_MapIDName2	ISOalpha2, MapIDNameAbrv ISOalpha3, MapIDNameAbrv ISOalpha2, MapIDName2 ISOname2, MapIDNameAbrv ISOalpha3, MapIDName2 ISOname2, MapIDName2

* Some of the default lookup data set names can be replaced with the data set name of your choice. CITY, STREET, ZIP, and ZIP+4 geocoding methods allow this data set name replacement.

** The default index variable can be replaced with an alternate variable name. CITY, STREET, ZIP, and ZIP+4 geocoding methods allow this variable name replacement.

Loading Data Sets into Memory

You can load your lookup data sets into memory by using the SASFILE statement. Loading data into memory reduces I/O processing and can improve the speed of your geocoding operation. You should test your geocoding operations with the lookup data sets loaded into memory to determine whether there is sufficient memory and whether your performance is increased.

For more information, see "SASFILE Statement" in *SAS Statements: Reference* in the *SAS Statements: Reference*.

Minimizing the Use of a Network

When geocoding large numbers of addresses, network issues can affect run times. You can obtain faster run times occur if the lookup data sets are stored locally, and if they are stored on an internal disk rather than on an external drive.

However, system administrators sometimes install the large lookup data sets in a central location that can be accessed only over the network. In that case, you can obtain faster run times by geocoding large amounts of data during times of less network traffic.

Avoiding the Use of CEDA When Accessing Lookup Data Sets

The cross-environment data access (CEDA) feature enables a SAS file that was created in a directory-based operating environment such as UNIX or Windows to be processed in an incompatible environment. It could also be processed under an incompatible session encoding. With CEDA, the processing is automatic and transparent.

Because CEDA does not support the use of indexes, it can negatively affect geocoding. This is not a concern when you geocode a small number of addresses. However, it does impede larger geocoding runs.

For best performance, the lookup data sets should reside on the same operating system used for geocoding. You can either install the lookup data sets there initially or use either CPORT or CIMPORT to move them. If you import them, make sure that the appropriate data set indexes are preserved or rebuilt.

Excluding the WORK Library from Microsoft Security Essentials and Microsoft Forefront Antivirus Programs

You might use Microsoft Security Essentials or Microsoft Forefront as your antivirus program. If you do, then you need to add the location for the WORK library to the list of excluded files and locations in Microsoft Security Essentials or Microsoft Forefront. The Microsoft 32- and 64-bit operating systems affected are Windows Vista, all editions of Windows 7, Windows XP Professional, all editions of Windows Server 2003, and Windows for 64-Bit Itanium-based systems.

To add the location for the WORK library to the excluded files and locations list in Microsoft Security Essentials, perform the following steps:

1. Click the **Settings** tab in Microsoft Security Essentials.
2. Select **Excluded files & locations**.
3. Click **Add**, and navigate to and select the directory location of the WORK library.
4. Click **OK**, and then click **Save changes**.

For information about how to exclude files and folders in Microsoft Forefront, visit this Forefront link, Excluding files, folders, and file types from scans, or contact Microsoft Technical Support directly.

Note: If you do not add the WORK library to the excluded files and locations list in Microsoft Security Essentials or Microsoft Forefront, then the following errors might occur:

```
ERROR: Rename of temporary member for WORK. failed
```

```
ERROR: User does not have appropriate authorization level for library WORK
```

See Also

"IBUFSIZE= System Option" in *SAS System Options: Reference*

Syntax: GEOCODE Procedure

PROC GEOCODE DATA=*address-data-set* <*option(s)*>;

PROC GEOCODE Statement

Identifies the data set that contains the address data that you want to geocode and optionally, a lookup data set. You can also specify an output data set, the geocoding method, alternate names for geocoding variables, and additional attribute variables to associate with the matched addresses.

Syntax

PROC GEOCODE DATA=*address-data-set* <*option(s)*>;

Summary of Optional Arguments

ADDRESSCITYVAR=*character-variable*
: specifies the character variable in the input address data set that contains the city names.

ADDRESSCOUNTRYVAR=*character-variable*
: specifies the character variable in the input address data set that contains the country name or identifier for international CITY geocoding.

ADDRESSPLUS4VAR=*variable*
: specifies the variable in the input address data set that contains ZIP+4 extensions.

ADDRESSSTATEVAR=*character-variable*
: specifies the character variable in the input address data set that contains the two-character postal abbreviation for each state or province.

ADDRESSVAR=*variable*
: specifies the address variable for STREET, CUSTOM, or RANGE geocoding.

ADDRESSZIPVAR=*variable*
: specifies the variable in the input address data set that contains the five-digit ZIP code values or non-U.S. postcode values.

ATTRIBUTEVAR=(*variable-1*, *variable-2*, …)
: lists non-geocoding variables in the lookup data set that are to be added to the output data set.

BEGINRANGEVAR=*variable*
: specifies the numeric variable in your range data set that contains the beginning IP address for each range of addresses.

DIRECTION=*data-set*
: specifies an alternate data set of street direction names.

ENDRANGEVAR=*variable*
: specifies the numeric variable in your range data set that contains the ending IP address for each range of addresses.

LOOKUP=*lookup-data-set*
: specifies a SAS data set that associates coordinates with addresses for specific geocoding methods.

LOOKUPCITY=*city-matching-data-set*
: specifies the city matching data set for associating coordinates with addresses when performing CITY geocoding.

LOOKUPCITYVAR=*character-variable*
: specifies the character variable in the lookup data set that contains the city names.

LOOKUPCOUNTRYVAR=*character-variable*
: specifies the character variable in the lookup data set that contains the country name or identifier.

LOOKUPKEYVAR=*variable*
: specifies the key variable for the lookup data set that links into the RANGE data set.

LOOKUPLATVAR=*numeric-variable*
: specifies the numeric variable in the lookup data set that contains the latitude of the geocoded location.

LOOKUPLONGVAR=*numeric-variable*
: specifies the numeric variable in the lookup data set that contains the longitude of the geocoded location.

LOOKUPPLUS4VAR=*variable*
: specifies the variable in the lookup address data set that contains ZIP+4 extensions.

LOOKUPSTATEVAR=*character-variable*
: specifies the character variable in the lookup data set that contains the two-character postal abbreviation for the states or provinces.

LOOKUPSTREET=*street-matching-data-set*
: specifies the street matching data set for associating coordinates with addresses when performing STREET geocoding.

LOOKUPVAR=*variable*
: specifies the variable in the lookup data set that contains non-address values used for geocoding with CUSTOM regions.

LOOKUPXVAR=*numeric-variable*
: specifies the numeric variable in the lookup data set that contains the longitude of the geocoded location.

LOOKUPYVAR=*numeric-variable*
: specifies the numeric variable in the lookup data set that contains the latitude of the geocoded location.

LOOKUPZIPVAR=*variable*
: specifies the variable in the lookup data set that contains the five-digit ZIP code values or the non-U.S. postcode values.

METHOD=*geocoding-method*
: specifies the primary geocoding method.

NOCITY
: disables the secondary matching attempt using the CITY geocoding method when STREET or ZIP code geocoding does not find a match.

NOSTIMER

disables the informational messages sent to the SAS log that tracks the progress of the geocoding operation.

NOZIP
disables the secondary matching attempt using the ZIP geocoding method when PLUS4 or STREET geocoding does not find a match.

OUT=*output-data-set*
specifies a data set for the geocoded addresses.

RANGEDATA=*data-set*
specifies a data set that associates ranges of IPv4 addresses with locations.

RANGEDECIMAL
specifies that the values of the ADDRESSVAR= variable are in decimal or hexadecimal format.

RANGEKEYVAR=*variable*
specifies the key variable for the IPv4 range data set.

STATE=*data-set*
specifies a SAS data set that is used by the STREET geocoding method to equate state or province postal abbreviations with complete names.

TYPE=*data-set*
specifies a SAS data set that is used by the STREET geocoding method to standardize common street address elements.

Required Argument

DATA=*address-data-set*
specifies the SAS data set that contains address observations that you want to geocode.

Default If you do not specify this option, then the most recently created SAS data set is used.

Optional Arguments

ADDRESSCITYVAR=*character-variable*
specifies the character variable in the input address data set that contains the city names.

Default CITY

ADDRESSCOUNTRYVAR=*character-variable*
specifies the character variable in the input address data set that contains the country name or identifier for international CITY geocoding. The variable value can contain either a two- or three-character country name abbreviation or the complete country name.

Defaults If your input country variable has a length of two characters, then the ISOALPHA2 variable in the lookup city data set is used by default for geocoding.

If your input variable has a length of three characters, then the ISOALPHA3 variable in the lookup city data set is used by default for geocoding.

Requirement This option is required for geocoding outside of the United States.

Notes The use of this option implies that you are geocoding non-U.S. locations.

When this option is specified, the default CITY lookup data set is MAPSGFK.WORLD_CITIES. If this option is not used, the GEOCODE procedure assumes that you are geocoding U.S. locations and uses MAPSGFK.USCITY_ALL as the default lookup data set.

Tip In some cases this option can reference an input data set variable that contains complete country names rather than abbreviated ISO values. In these cases use the option "LOOKUPCOUNTRYVAR=*character-variable*" on page 138 to specify the associated variable in your lookup data set.

ADDRESSPLUS4VAR=*variable*

specifies the variable in the input address data set that contains ZIP+4 extensions. The variable can be either numeric or character, but it must be the same type as the ZIP+4 variable in the lookup data set (LOOKUPPLUS4VAR=).

Default PLUS4

Note The variables that you specify for ADDRESSPLUS4VAR= and ADDRESSZIPVAR= must be the same data type.

ADDRESSSTATEVAR=*character-variable*

specifies the character variable in the input address data set that contains the two-character postal abbreviation for each state or province.

This variable can contain the two-character state abbreviation (for example, NY). It can also contain the complete state name (for example, New York).

Default STATE

ADDRESSVAR=*variable*

specifies the address variable for STREET, CUSTOM, or RANGE geocoding. For STREET geocoding, specifies the variable in the address data set that contains the street address values (for example, "1229 North Main St.")

For CUSTOM and RANGE geocoding, the ADDRESSVAR= option specifies the variable in the address data set that contains non-address input values. The variable can be character or numeric. The variable specified can contain either an IPv4 address or an IPv6 address.

Default For STREET geocoding, the default name is ADDRESS.

Interactions For the CUSTOM method of geocoding, this option is used in conjunction with the LOOKUPVAR= option to geocode with unconventional values. Examples of unconventional variable values include internal sales territories, Metropolitan Statistical Areas (MSA), and Internet Protocol (IP) addresses.

For the RANGE method of geolocating IP addresses, when this option is used in conjunction with the RANGEDECIMAL option, the ADDRESSVAR= variable values must be in decimal form for IPv4. For IPv6 the format is either decimal or hexadecimal. The colon separated IPv6 addresses are hexadecimal, but the up to 48 digit numbers are decimal. IPv6 is supported by range geolocating.

ADDRESSZIPVAR=*variable*
specifies the variable in the input address data set that contains the five-digit ZIP code values or non-U.S. postcode values.

The variable can be either numeric or character, but it must be the same type as the ZIP code variable in the lookup data set (specified by the LOOKUPZIPVAR= option).

Default ZIP

Restriction The values for the ZIP code variable must be five digits. You can use the `Z5.` format to prepend leading zeros to any ZIP code values that have fewer than five digits.

Note Postcodes from other countries can also be used if the appropriate lookup data is imported into a SAS data set. See "Non-U.S. Postcodes" on page 118 for more information.

ATTRIBUTEVAR=(*variable-1*, *variable-2*, …)
lists non-geocoding variables in the lookup data set that are to be added to the output data set. The values are based on the geocoded location. Examples include county, census block, and time zone. Variable names can be separated by commas or spaces.

Notes The values for additional attribute variables are not added to observations in the output data set where the match type is "City mean" or "ZIP mean".

If an attribute variable has the same name as a variable in the address data set, then that attribute variable is not added to the output data set.

For the STREET geocoding method, only attribute variables from the street segment lookup data set can be included.

Only attribute values from the primary lookup data set that are associated with the specified geocoding method can be added to the output data set. For example, with the STREET geocoding method, you cannot specify that attribute variables be added from the ZIP or CITY lookup data sets. To add attribute variables from the ZIP lookup data set, you must run the GEOCODE procedure a second time. This second run uses the ZIP method on the addresses for which no STREET match was found.

Example ATTRIBUTEVAR=(STATENAME, COUNTYNM)

BEGINRANGEVAR=*variable*
specifies the numeric variable in your range data set that contains the beginning IP address for each range of addresses.

DIRECTION=*data-set*
specifies an alternate data set of street direction names. A data set named SASHELP.GCDIRECT that contains text strings indicating direction is installed. These can be used in U.S. street name prefixes and suffixes and text strings such as northwest or NW. If the %GEOBASE2GEOCODE macro program is used to import Canadian street data, the data set GCDIRECT_CAN is created with English and French directional text strings. Use the DIRECTION= option to reference this alternate data set when geocoding Canadian street addresses.

Aliases DIR=

DIRECT=

Default SASHELP.GCDIRECT (U.S. street geocoding only)

Restrictions The specified data set should contain two variables named DIRABRV and DIRECTION. Each variable is capable of containing text strings, but the DIRECTION variable must contain only alphabetic characters.

A text string indicating direction must exist in the DIRECTION parameter in the specified data set. This enables successful street matching of addresses using this text string. For example, a match for street address "300 N Academy St" cannot be found if the DIRECTION parameter does not include a text string of "N". This is true even if the DIRABRV parameter in the data set contains this "N" text string.

ENDRANGEVAR=*variable*
specifies the numeric variable in your range data set that contains the ending IP address for each range of addresses.

LOOKUP=*lookup-data-set*
specifies a SAS data set that associates coordinates with addresses for specific geocoding methods. When the geocoding method is ZIP, RANGE, or CUSTOM, the data set is searched for observations that match the address observations. The variables that are required for your lookup data set depend on your geocoding method. See "About Alternate ZIP Code Lookup Data" on page 117.

You can specify that non-geocoding variables such as COUNTY, TIME ZONE, and AREA CODE from the lookup data set be added to the output data set. Do this by using the ATTRIBUTEVAR= option in the PROC GEOCODE statement.

Default For the ZIP geocoding method, the default lookup data set SASHELP.ZIPCODE is used.

Restriction This option is not valid when using the CITY geocoding method. Use the LOOKUPCITY= option instead.

Requirements For the IPv4 or IPv6 RANGE or the CUSTOM geocoding method, you must specify the lookup data set with the LOOKUP= option. IPv6 data set names must have the number 6 appended.

You specify the CITY method lookup data set with the LOOKUPCITY= option.

See "LOOKUPCITY=*city-matching-data-set*" on page 137

LOOKUPCITY=*city-matching-data-set*
specifies the city matching data set for associating coordinates with addresses when performing CITY geocoding. Default lookup data sets are used by the GEOCODE procedure without having to specify this option. Use this option to specify an alternate lookup data set. The variables that are required in an alternate lookup data set depend on whether all of your addresses are within the United States or if there are non-U.S. addresses. See "About Alternate U.S. City Lookup Data" on page 112 or "About Alternate World City Lookup Data" on page 112 for a list of required variables.

The city lookup data set can also contain other attribute variables (such as ISONAME and COUNTY_NAME.You can add attribute values to the lookup data set using the "ATTRIBUTEVAR=(*variable-1*, *variable-2*, …)" on page 136option.

Default For the CITY geocoding method, when the ADDRESSCOUNTRYVAR= option is not specified, the GEOCODE procedure assumes that you are geocoding only U.S. locations and uses MAPSGFK.USCITY_ALL as the default U.S. city lookup data set. When the ADDRESSCOUNTRYVVAR= option is specified, the GEOCODE procedure assumes that you are geocoding worldwide locations and uses MAPSGFK.WORLD_CITIES as the default international city lookup data set.

Restrictions This option is not valid with the RANGE or CUSTOM geocoding method.

You specify the alternate lookup data set with the LOOKUPCITY= option when using the CITY geocoding method.

LOOKUPCITYVAR=*character-variable*

specifies the character variable in the lookup data set that contains the city names. The lookup data set used by the GEOCODE procedure determines which default city name variable is used.

Defaults CITY with either a SASHELP.ZIPCODE or a user-created lookup data set. (The variable values do not have to be normalized.)

CITY2 when the lookup data set is MAPSGFK.USCITY_ALL, MAPSGFK.WORLD_CITIES, or MAPSGFK.WORLD_CITIES_ALL. (The CITY2 variable values in these data sets are normalized. This means that they are converted to uppercase and stripped of all spaces and characters that are not alphabetic or numeric.)

Requirement This option must be specified when you include this type of variable in a lookup data set not supplied by SAS and you intend to geocode with that variable.

Note The CITY variable value in the MAPSGFK.USCITY_ALL, MAPSGFK.WORLD_CITIES, and MAPSGFK.WORLD_CITIES_ALL lookup data sets contains the mixed case version of the city name.

LOOKUPCOUNTRYVAR=*character-variable*

specifies the character variable in the lookup data set that contains the country name or identifier. Use this option for international CITY geocoding.

Default The default variables are either two- or three-character country name abbreviations in the MAPSGFK.WORLD_CITIES_ALL lookup data set. If the ADDRESSCOUNTRYVAR= option references a two-character country abbreviation (such as 'GB') in the input address data set, the default lookup country variable name is ISOALPHA2. If the ADDRESSCOUNTRYVAR= refers to a three-character country name ('GBR'), the default lookup country variable is ISOALPHA3.

Requirement This option must be specified when you include this type of variable in a lookup data set not supplied by SAS and you intend to geocode with that variable.

Note This option can also reference a lookup data set variable containing complete country names instead of abbreviations.

See "ADDRESSCOUNTRYVAR=*character-variable*" on page 134

LOOKUPKEYVAR=*variable*
specifies the key variable for the lookup data set that links into the RANGE data set. The values of the key variable correspond to values in the variable that you specify for the RANGEKEYVAR= option. The key variable provides a link from the RANGEDATA= data set to the lookup data set. This key variable is used to map more than one IPv4 address range to one location.

Restrictions The data type of the key variable must match the variable that you specify for the RANGEKEYVAR= option.

This option is used only with IPv4 address data. The geocoder uses the key variable when comparing the two lookup data sets, CITYBLOCKS and CITYLOCATION, both of which contain the key variable.

LOOKUPLATVAR=*numeric-variable*
specifies the numeric variable in the lookup data set that contains the latitude of the geocoded location. The lookup data set used by the GEOCODE procedure determines which default latitude variable is used.

Defaults The LAT variable is the default when the GEOCODE procedure uses the MAPSGFK.USCITY_ALL, MAPSGFK.WORLD_CITIES, or the MAPSGFK.WORLD_CITIES_ALL lookup data set.

The Y variable is the default when SASHELP.ZIPCODE or a user-created lookup data set is used.

Requirement This option must be specified when you include this type of variable in a lookup data set not supplied by SAS and you intend to geocode with that variable.

Notes This option supersedes the option LOOKUPYVAR=. Use this option instead of the LOOKUPYVAR= option.

The latitude variable name that you specify with this option must be contained in all of the lookup data sets that might be used during your geocoding run. For example, when you specify LOOKUPLATVAR=LATITUDE with the ZIP and CITY geocoding methods, the latitude variable in your ZIP and CITY lookup data sets must contain that exact variable name. You cannot use LAT in one lookup data set and Y in another.

LOOKUPLONGVAR=*numeric-variable*
specifies the numeric variable in the lookup data set that contains the longitude of the geocoded location. The lookup data set used by the GEOCODE procedure determines which default longitude variable is used.

Defaults The LONG variable is the default when the GEOCODE procedure uses the MAPSGFK.USCITY_ALL, MAPSGFK.WORLD_CITIES, or the MAPSGFK.WORLD_CITIES_ALL lookup data set.

The X variable is the default when SASHELP.ZIPCODE or a user-created lookup data set is used.

Requirement This option must be specified when you include this type of variable in a lookup data set not supplied by SAS and you intend to geocode with that variable.

Notes This option supersedes the option LOOKUPXVAR=. Use this option instead of the LOOKUPXVAR= option.

The longitude variable name that you specify with this option must be contained in all of the specified lookup data sets that might be used during your geocoding run. For example, when you specify LOOKUPLONGVAR=LONGITUDE with the ZIP and CITY geocoding methods, the longitude variable in your ZIP and CITY lookup data sets must contain that exact variable name. You cannot use LONG in one lookup data set and X in another.

LOOKUPPLUS4VAR=*variable*

specifies the variable in the lookup address data set that contains ZIP+4 extensions. The variable can be either numeric or character, but it must be the same type as the ZIP+4 variable in the input address data set (ADDRESSPLUS4VAR=).

Default PLUS4

Requirement This option must be specified when you include this type of variable in a lookup data set not supplied by SAS and you intend to geocode with that variable.

LOOKUPSTATEVAR=*character-variable*

specifies the character variable in the lookup data set that contains the two-character postal abbreviation for the states or provinces.

Default STATECODE

Requirement This option must be specified when you include this type of variable in a lookup data set not supplied by SAS and you intend to geocode with that variable.

LOOKUPSTREET=*street-matching-data-set*

specifies the street matching data set for associating coordinates with addresses when performing STREET geocoding. The GEOCODE procedure expects the street matching data set to have a name that ends with M. The library must also contain two associated data sets that share the same base name as the M data set. The difference is that one name ends with an S (segment) and the other name ends with a P (coordinate). For example, if you specify the street matching data set MYMAPS.GEORGIAM, then the MYMAPS library must also contain the GEORGIAS and GEORGIAP data sets.

For more information about the data sets for STREET geocoding, see "About Street Lookup Data" on page 94.

Default The SASHELP.USM data set, which is not installed with SAS. You can download the USM, USS, and USP data sets which cover the entire United

States and its overseas territories from the SAS Maps Online web site at http://support.sas.com/rnd/datavisualization/mapsonline/index.html.

LOOKUPVAR=*variable*
specifies the variable in the lookup data set that contains non-address values used for geocoding with CUSTOM regions. The variable can be character or numeric. For the CUSTOM method of geocoding, this option is used in conjunction with the ADDRESSVAR= option to geocode with unconventional values. Examples of unconventional variable values include internal sales territories, Metropolitan Statistical Areas (MSA), and telephone area codes.

Restriction This option is used exclusively with the CUSTOM geocoding method.

LOOKUPXVAR=*numeric-variable*
specifies the numeric variable in the lookup data set that contains the longitude of the geocoded location.

Default LONG

Requirement This option must be specified when you include this type of variable in a lookup data set not supplied by SAS and you intend to geocode with that variable.

Note This option is superseded by the option LOOKUPLONGVAR=. If you specify the LOOKUPXVAR= option, a message in the SAS log directs you to start using the LOOKUPLONGVAR = option.

LOOKUPYVAR=*numeric-variable*
specifies the numeric variable in the lookup data set that contains the latitude of the geocoded location.

Default LAT

Requirement This option must be specified when you include this type of variable in a lookup data set not supplied by SAS and you intend to geocode with that variable.

Note This option is superseded by the option LOOKUPLATVAR=. If you specify the LOOKUPYVAR= option, a message in the SAS log directs you to start using the LOOKUPLATVAR = option.

LOOKUPZIPVAR=*variable*
specifies the variable in the lookup data set that contains the five-digit ZIP code values or the non-U.S. postcode values.

The variable can be either character or numeric, but it must be the same type as the ZIP code variable in the input address data set (ADDRESSZIPVAR=).

Default ZIP

Restriction The values for a character ZIP code variable must be five digits. You can use the `Z5.` format to prepend leading zeros to any ZIP code values that have fewer than five digits.

Requirement This option must be specified when you include this type of variable in a lookup data set not supplied by SAS and you intend to geocode with that variable.

Note Postcodes from other countries can also be used if the appropriate lookup data is imported into a SAS data set. See "Non-U.S. Postcodes" on page 118 for more information.

METHOD=*geocoding-method*

specifies the primary geocoding method. This parameter is optional. Specify one of the following:

CITY

specifies the CITY geocoding method. The GEOCODE procedure attempts to match certain variables from your address data set with the lookup data set. The procedure uses city and state variables for U.S. city matching, and city and country variables for world city matching. When using a world city lookup data set, you have the option of specifying a state or province variable to help the procedure match cities. A _MATCHED_ value of **`State mismatch`** indicates that the geocoder found a city and country match but the state or province name was not matched. Separate city, country, and state variables are required in the address and lookup data sets.

Notes The city and state matching methods are case-insensitive.

When the GEOCODE procedure uses SASHELP.ZIPCODE as the CITY lookup data set and finds multiple matches for an address, the procedure averages the latitude and longitude values. These values are written to the output data set as Y and X coordinates. For lookup data sets provided by GfK GeoMarketing, the coordinates appear in the output data set as LAT and LONG. If multiple cities are matched in the GfK lookup data sets, the latitude and longitude values are set to missing and the _MATCHED_ value is the number of cities matched.

CUSTOM

specifies the CUSTOM geocoding method. The GEOCODE procedure attempts to match custom variables that you specify by using the LOOKUPVAR= and ADDRESSVAR= options. Examples of custom variables include internal sales territories and Metropolitan Statistical Areas (MSA).

Requirement You must use the ADDRESSVAR= option to identify the address variable in your address data. You must also use the LOOKUPVAR= option to identify the non-address variable in the lookup data set.

PLUS4

specifies the PLUS4 geocoding method. The GEOCODE procedure attempts to match the five-digit ZIP code and ZIP+4 extension from your address data set with the lookup data set.

If no match is found, then the GEOCODE procedure attempts to match the five-digit ZIP code only. If multiple ZIP matches are found, then the matching latitude and longitude coordinate values are averaged.

If no ZIP+4 or ZIP matches are found, then the GEOCODE procedure performs CITY geocoding. If multiple CITY matches are found in the SASHELP.ZIPCODE or user-supplied lookup data set, then the matching latitude and longitude coordinate values are averaged.

Interaction You can disable the secondary ZIP code matching by using the NOZIP option. You can disable the secondary CITY matching by using the NOCITY option.

RANGE
specifies the RANGE geocoding method. The GEOCODE procedure attempts to match an IP address from your address data set to a range of IP addresses from the range data set. If a match is found, then a key variable is used to match the IP address to a set of coordinates in the lookup data set.

STREET
specifies the STREET geocoding method. The GEOCODE procedure first attempts to match the street name and ZIP or postal code. If no match is found, then the GEOCODE procedure attempts to match the street name, city name, and two-character state abbreviation or full state name. If the second match fails, then the ZIP method and the CITY method are used instead.

If a street match is found, longitude and latitude coordinate values are interpolated along the street by using the house number from your input address.

For more information, see "Understanding Street Geocoding" on page 94.

Interactions You can disable the secondary ZIP matching by using the NOZIP option.

You can disable the secondary CITY matching by using the NOCITY option.

ZIP
specifies the ZIP code geocoding method. The GEOCODE procedure attempts to match the five-digit ZIP code or non-U.S. postal code from your address data set with the lookup data set. If no match is found, then the CITY method is used instead.

Interaction You can disable the secondary matching by using the NOCITY option.

Default ZIP

Requirement For a list of options that are required for each geocoding method, see "Required and Optional Arguments for Geocoding" on page 147.

Interaction If you specify more than one method, then the GEOCODE procedure uses the last method that you specify as the primary geocoding method.

NOCITY
disables the secondary matching attempt using the CITY geocoding method when STREET or ZIP code geocoding does not find a match.

Default If the ZIP code geocoding method does not find a match, then the GEOCODE procedure attempts to match the city and state values using the CITY geocoding method. The procedure does the same calculation if the STREET geocoding method does not find a match for the street address or ZIP code.

Restriction Cannot be used with the CITY geocoding method.

NOSTIMER

disables the informational messages sent to the SAS log that tracks the progress of the geocoding operation. If the input data set includes 1,000 or more observations, then the GEOCODE procedure writes periodic messages to the SAS log. The messages show the percentage of addresses completed and the estimated time remaining. This option disables those messages.

Note Unbuffered, real-time log output is useful in monitoring the progress of large geocoding runs. The GEOCODE procedure can write messages to the logon real time only if the LOGPARM system option was set to WRITE=IMMEDIATE when the SAS session was invoked. In addition, you must be using SAS in batch mode or line mode. Setting LOGPARM="WRITE=IMMEDIATE" causes messages to be written immediately to the SAS log rather than buffered for later output. If you are using the SAS Windowing environment, you can use the ALTLOG system option to specify a log file. In this case, only the external ALTLOG file is written immediately. The Log window output is buffered in all cases. For more information about the LOGPARM option, see "LOGPARM= System Option" in *SAS System Options: Reference*.

NOZIP

disables the secondary matching attempt using the ZIP geocoding method when PLUS4 or STREET geocoding does not find a match.

Default If PLUS4 or STREET geocoding do not find a match, then the GEOCODE procedure attempts to match the five-digit ZIP code or non-U.S. postal code.

Restriction Cannot be used with the ZIP geocoding method.

Note If your data set contains many missing ZIP+4 values, then the NOZIP option might improve performance.

Tip This option should be specified when geocoding with non-U.S. lookup data sets that do not contain postal codes.

OUT=*output-data-set*

specifies a data set for the geocoded addresses. All of the variables in the input address data set are copied to the output data set. Also added to the output data set are the following:

- Longitude and latitude variables for the geocoded location of the match
- optional variables, if specified by the ATTRIBUTEVAR option, that contain additional information about the geocoded location
- a variable named _MATCHED_ indicating how the match was made
- Additional variables written to the output data set for STREET geocoding are M_ADDR, M_CITY, M_STATE, M_ZIP, M_OBS, _STATUS_, _NOTES_, and _SCORE_. See "Output Variables for Street Geocoding" on page 100 for details.

If the output data set name that you specify already exists, then it is replaced without warning. If the output data set is the same as the input data set, then the input data set is updated by the geocoding operation.

If you omit the OUT= option, then the name of the output data set is DATA*n*, where *n* is the smallest integer that produces a unique name. For example, if a DATA1 data set exists, then the default name for the output data set is DATA2.

If the input address data set has no label, then the label of the output data set contains the text, "geocoded **`date`**". **`Date`** refers to the date on which the output was created. If the input address data set has a label, then that label is placed on the output data set and the "geocoded **`date`**" text is appended to it.

Tip You can use standard SAS data set options with the OUT= option. For example, the following code specifies which variables to keep from the input data set.

```
proc geocode data=address
   out=geocoded (keep=name city state zip);
```

For more information about data set options, see *SAS Data Set Options: Reference*.

RANGEDATA=*data-set*

specifies a data set that associates ranges of IPv4 addresses with locations. The data set should contain variables that identify the starting IPv4 address, ending IPv4 address, and location ID for each range of IPv4 addresses. Use the LOOKUP= option to specify the data set with the latitude and longitude coordinates for each location corresponding to the IPv4 addresses.

Restriction If you are providing the geocoder with an IPv6 data set and an IPv4 data set, they must reside in the same directory location.

Notes There are no default data set names. You must specify all data set names. Variable names read from the header record in the IPv4 MaxMind CSV files are used as the default names.

The RANGEDATA= option does not apply to IPv6 addresses. Use the LOOKUP= option to specify a data set with IPv6 addresses. This data set contains the beginning and ending IPv6 addresses, the location ID for each range of IPv6 addresses, and the latitude and longitude coordinates for each location.

RANGEDECIMAL

specifies that the values of the ADDRESSVAR= variable are in decimal or hexadecimal format. IPv6 addresses in hexadecimal format are supported.

Default The IPv4 addresses in the ADDRESSVAR= variable are in dotted quad notation. For example, the IP address 192.168.0.1 is represented as 3232235521 in decimal form. The IPv6 addresses in the ADDRESSVAR= variable are in eight groups of four hexadecimal digits separated by colons. An example of a full IPv6 address is FE80:0000:0000:0000:0202:B3FF:FE1E:8329. Collapsed forms of IPv6 addresses are also accepted. The collapsed format for the previous example is FE80::0202:B3FF:FE1E:8329. The full IPv6 address is also represented as the decimal number 338288524927261089654163772891438416681 in the IPv6 lookup data.

See "Internet Protocol Version 6 (IPv6)" in *SAS Language Reference: Concepts*

RANGEKEYVAR=*variable*

specifies the key variable for the IPv4 range data set. The values of the key variable correspond to values in the variable that you specify for the LOOKUPKEYVAR= option. They also provide a link into the LOOKUP= data set. This key variable is used to map more than one IPv4 address range to one location.

Restrictions The data type of the key variable must match the variable that you specify for the LOOKUPKEYVAR= option.

This option is used only with IPv4 address data. The geocoder uses the key variable when comparing the two lookup data sets, CITYBLOCKS and CITYLOCATION, both of which contain the key variable.

STATE=*data-set*

specifies a SAS data set that is used by the STREET geocoding method to equate state or province postal abbreviations with complete names. For example, use this option to specify a customized state data set that contains alternate state or province names.

Default SASHELP.GCSTATE

Notes The GCSTATE data set installed in SASHELP includes U.S. states and Canadian provinces. Observations for other countries can be added.

The variables MapIDName2 and ISOname2 are normalized copies of MapIDName and ISOname, respectively. Normalizing converts to uppercase the original value and removes all punctuation and spaces to provide a clean text string. This enables faster geocoding.

TYPE=*data-set*

specifies a SAS data set that is used by the STREET geocoding method to standardize common street address elements. For example, the data set of street types might standardize "parkway", "prkwy" and "pkwy" to a standard form "pkwy" to facilitate matching. Other examples include avenue (AVE) and boulevard (BLVD).

Default The SASHELP.GCTYPE data set.

Restriction All values in SASHELP.GCTYPE must be converted to uppercase and contain no punctuation or spaces.

Notes When adding street types to the data set, be sure to add at least one observation for each new type. Add two observations if the type can be abbreviated. In the first observation, the NAME value contains the full name of the type and the TYPE value contains the abbreviation. The second observation puts the abbreviation in both the NAME and TYPE values. For example, because 'AVENUE' can be abbreviated, its two observations are added as follows:

```
NAME='AVENUE' and TYPE='AVE'

NAME='AVE' and TYPE='AVE'
```

Some types are short enough that they have no abbreviation. In those cases, only one entry is needed. Both variables contain the full length name. For example:

```
NAME='WAY' and TYPE='WAY'
```

If the %GEOBASE2GEOCODE macro program is used to import Canadian street data, it creates the GCTYPE_CAN data set. This data set contains Canadian street types and abbreviations. Use the TYPE= option to specify this alternate data set when geocoding Canadian street addresses.

Details

Converting SAS/GIS Programs

You might have existing SAS/GIS batch geocoding programs that use the %GCBATCH autocall macro. To convert these programs to the GEOCODE procedure, using the option name from this macro is an acceptable alias in most situations. For more information, see the *SAS/GIS: Spatial Data and Procedure Guide*.

Required and Optional Arguments for Geocoding

Depending on the geocoding method that you use, additional options might also be required. The following table specifies the required and optional arguments for each method:

Table 5.3 *Required and Optional Arguments for Each Geocoding Method*

Geocoding Method	Arguments
CITY	Non-U.S. CITY geocoding requires the following options: • ADDRESSCOUNTRYVAR= • ADDRESSCITYVAR= • LOOKUPCITYVAR= These options are optional when geocoding cities within the United States. In addition, U.S. and non-U.S. CITY geocoding can use the following options: • ADDRESSSTATEVAR= • LOOKUPCITY= • LOOKUPCOUNTRYVAR= *Note:* This option is required when geocoding cities outside of the United States. • LOOKUPLATVAR= • LOOKUPLONGVAR= • LOOKUPSTATEVAR= If your data does not use the default variable names for any of these options, then you must specify the correct variable names with the appropriate options.
CUSTOM	CUSTOM geocoding requires the following: • ADDRESSVAR= • LOOKUPCITY= • LOOKUPVAR= If your lookup data set does not use the default variable names for X and Y, then the following options are also required: • LOOKUPLATVAR= • LOOKUPLONGVAR=

Geocoding Method	Arguments
PLUS4	PLUS4 geocoding requires the LOOKUP= argument. PLUS4 geocoding can use the following options: • ADDRESSPLUS4VAR= • ADDRESSZIPVAR= • LOOKUPLATVAR= • LOOKUPLONGVAR= • LOOKUPPLUS4VAR= • LOOKUPZIPVAR= If your data does not use the default variable names for any of these options, then you must specify the correct variable names with the appropriate options.
RANGE	RANGE geocoding requires the following: • ADDRESSVAR= • BEGINRANGEVAR= • ENDRANGEVAR= • LOOKUP= • LOOKUPKEYVAR= • RANGEDATA= • RANGEKEYVAR= RANGE geocoding can use the following options: • LOOKUPLATVAR= • LOOKUPLONGVAR= If your data does not use the default variable names for any of these options, then you must specify the correct variable names with the appropriate options. If your IPv4 address values are in decimal format, then you must also specify the RANGEDECIMAL option.

Geocoding Method	Arguments
STREET	STREET geocoding can use the following options: • ADDRESSCITYVAR= • ADDRESSSTATEVAR= • ADDRESSZIPVAR= • ADDRESSVAR= • DIRECTION= • LOOKUPCITYVAR= • LOOKUPLATVAR= • LOOKUPLONGVAR= • LOOKUPSTATEVAR= • LOOKUPSTREET= • LOOKUPZIPVAR= • STATE= • TYPE= If your data does not use the default variable names for any of these options, then you must specify the correct variable names with the appropriate options. The following options are not required if you specify the NOCITY option: • ADDRESSCITYVAR= • ADDRESSSTATEVAR= • LOOKUPCITYVAR= • LOOKUPSTATEVAR= The following options are not required if you specify the NOZIP option: • ADDRESSZIPVAR= • LOOKUPZIPVAR=

Geocoding Method	Arguments
ZIP	ZIP geocoding can use the following options: • ADDRESSCITYVAR= • ADDRESSSTATEVAR= • ADDRESSZIPVAR= • LOOKUPCITYVAR= • LOOKUPLATVAR= • LOOKUPLONGVAR= • LOOKUPSTATEVAR= • LOOKUPZIPVAR= If your data does not use the default variable names for any of these options, then you must specify the correct variable names with the appropriate options. The following options are not required if you specify the NOCITY option: • ADDRESSCITYVAR= • ADDRESSSTATEVAR= • LOOKUPCITYVAR= • LOOKUPSTATEVAR=

Examples: GEOCODE Procedure

Example 1: Geocoding Using Default Values

Features: ZIP geocoding method
Procedure options
OUT=

Other features: Base SAS functions
SAS DATA step
PRINT procedure

Data set: SASHELP.ZIPCODE (lookup data set)

Sample library member: GEOSMPL

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

This example shows the simplest form of the GEOCODE procedure, specifying only the OUT= option. The GEOCODE procedure compares the input data set to the lookup data and outputs any match that it finds based on a five-digit ZIP code. The ZIP method is the default.

The result of using all of the default values is that the following is true:

- The input address data set is the most recently created SAS data set (this example assumes that you have just created WORK.CUSTOMERS).
- The ZIP geocoding method is used.
- The lookup data set is SASHELP.ZIPCODE.
- No variables are added to the output data set other than the X and Y coordinates, and a _MATCHED_ variable indicating whether and how the match was made.

The following output from PROC PRINT shows the output data set after running the GEOCODE procedure. Notice that the following geocoding variables have been added:

- coordinate variables X and Y from the lookup data set (SASHELP.ZIPCODE).
- a variable named _MATCHED_. This variable indicates whether the location was found by matching ZIP codes or by matching City and State (or whether no location was found because no match was made).

  Your X and Y or M_OBS values might differ depending on the version of SASHELP.ZIPCODE that is installed at your site.

Output

The following output from the PRINT procedure shows the GEOCODED_CUSTOMERS output data set after running the GEOCODE procedure

Output 5.1 *The GEOCODED_CUSTOMERS Output Data Set*

Y	X	M_OBS	_MATCHED_	address	city	state	zip	cust_ID
34.0736	-118.399	12550	City	555 Junk Street	Beverly Hills	CA	99999	1
39.1500	-75.532	7610	ZIP	115 E. Water St	Dover		19901	2
39.0953	-75.570	7619	ZIP	760 Moose Lodge Road	Camden		19934	3
39.7366	-75.549	7585	ZIP	200 S. Madison Str	Wilmington	DE	19801	4
39.7317	-75.669	7592	ZIP	4701 Limestone Road	Wilmington		19808	5
39.7877	-75.961	7420	ZIP	2117 N 4th St	Oxford	PA	19363	6
.	.	.	None	1313 Mockingbird Lane	Delray	CC	.	7
38.7265	-75.081	7648	ZIP	133 Silver Lake Dr	Rehoboth Beach	DE	19971	8
38.9035	-75.432	7641	ZIP	11 SE Front Street	Milford	DE	19963	9
38.6411	-75.611	20665	City	402 Nylon Boulevard	Seaford	DE	.	10
39.2997	-75.605	19424	City	363 E Commerce St	Smyrna	DE	.	11
38.4663	-75.150	7650	ZIP	5595 Polly Branch Rd	Selbyville	DE	19975	12
38.4593	-75.053	7626	ZIP	1209 Coastal Highway	Fenwick Island	DE	19944	13
39.1509	-75.693	7633	ZIP	2899 Arthursville Rd	Hartly	DE	19953	14
.	.	.	None	41 Bramhall St			.	15
38.4564	-75.578	20323	City	9320 Old Racetrack Rd	Delmar	DE	.	16
39.2282	-75.666	7635	ZIP	281 W Commerce Str	Kenton		19955	17
39.6264	-75.850	8616	ZIP	211 Blue Ball Road	Elkton	MD	21921	18
39.0695	-75.567	7653	ZIP	3893 Turkey Point Rd	Woodside	DE	19980	19

Program

```
data CUSTOMERS (label="Customer data for geocoding");
```

```
infile datalines dlm='#';
length address $ 24 city $ 24 state $ 2;
input address    /* House number and street name */
      zip        /* Customer ZIP code (numeric)  */
      city       /* City name                    */
      state      /* State abbreviation           */
;
cust_ID = _n_;   /* Assign customer ID number    */
datalines;
555 Junk Street # 99999 # Beverly Hills # CA
115 E. Water St # 19901 # Dover #
760 Moose Lodge Road # 19934 # Camden #
200 S. Madison Str # 19801 # Wilmington # DE
4701 Limestone Road # 19808 # Wilmington #
2117 N 4th St # 19363 # Oxford # PA
1313 Mockingbird Lane # . # Delray # CC
133 Silver Lake Dr # 19971 # Rehoboth Beach # DE
11 SE Front Street # 19963 # Milford # DE
402 Nylon Boulevard # . # Seaford # DE
363 E Commerce St # . # Smyrna # DE
5595 Polly Branch Rd # 19975 # Selbyville # DE
1209 Coastal Highway # 19944 # Fenwick Island # DE
2899 Arthursville Rd # 19953 # Hartly # DE
41 Bramhall St # . #  #
9320 Old Racetrack Rd # . # Delmar # DE
281 W Commerce Str # 19955 # Kenton #
211 Blue Ball Road # 21921 # Elkton # MD
3893 Turkey Point Rd # 19980 # Woodside # DE
;
run;

proc geocode out=geocoded_customers;
run;

proc print data=geocoded_customers noobs;
run;
```

Program Description

Generate the CUSTOMERS input data set of addresses that the GEOCODE procedure uses.

```
data CUSTOMERS (label="Customer data for geocoding");
infile datalines dlm='#';
length address $ 24 city $ 24 state $ 2;
input address    /* House number and street name */
      zip        /* Customer ZIP code (numeric)  */
      city       /* City name                    */
      state      /* State abbreviation           */
;
cust_ID = _n_;   /* Assign customer ID number    */
datalines;
555 Junk Street # 99999 # Beverly Hills # CA
115 E. Water St # 19901 # Dover #
760 Moose Lodge Road # 19934 # Camden #
200 S. Madison Str # 19801 # Wilmington # DE
```

```
4701 Limestone Road # 19808 # Wilmington #
2117 N 4th St # 19363 # Oxford # PA
1313 Mockingbird Lane # . # Delray # CC
133 Silver Lake Dr # 19971 # Rehoboth Beach # DE
11 SE Front Street # 19963 # Milford # DE
402 Nylon Boulevard # . # Seaford # DE
363 E Commerce St # . # Smyrna # DE
5595 Polly Branch Rd # 19975 # Selbyville # DE
1209 Coastal Highway # 19944 # Fenwick Island # DE
2899 Arthursville Rd # 19953 # Hartly # DE
41 Bramhall St # . #  #
9320 Old Racetrack Rd # . # Delmar # DE
281 W Commerce Str # 19955 # Kenton #
211 Blue Ball Road # 21921 # Elkton # MD
3893 Turkey Point Rd # 19980 # Woodside # DE
;
run;
```

Run the GEOCODE procedure with the generated input data set. This example assumes that CUSTOMERS is the most recently generated input data set. The default lookup data set, SASHELP.ZIPCODE, is used. GEOCODE uses the default ZIP method to compare the input data set to the lookup data and match observations based on a five-digit ZIP code.

```
proc geocode out=geocoded_customers;
run;
```

Print the entire GEOCODED_CUSTOMERS output data set, suppressing the observation column.

```
proc print data=geocoded_customers noobs;
run;
```

Example 2: Adding Additional Variables to the Output Data Set

Features: ZIP geocoding method

Procedure Options
- METHOD=
- DATA=
- OUT=
- ATTRIBUTEVAR=

Other features: Base SAS functions
- SAS DATA step
- PRINT procedure

Sample library member: GEOVARS

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

This example illustrates using the ATTRIBUTEVAR= option to add additional variables (from the lookup data set) to the output data set. The example also illustrates using the DATA= option to specify an input address data set.

The following output from PROC PRINT shows the output data set after running the GEOCODE procedure. Notice that the following variables have been added to the output data set:

- coordinate variables X and Y from the lookup data set (SASHELP.ZIPCODE) for the geocoded locations. Your X and Y values might differ depending on the version of SASHELP.ZIPCODE that is installed.
- a variable named _MATCHED_. This variable indicates whether the location was found by matching ZIP codes or by matching City and State (or whether no location was found because no match was made).
- an attribute variable named STATENAME from the lookup data set for each geocoded location. The lookup data set contains the full name of the state or territory .
- an attribute variable named COUNTYNM from the lookup data set for each geocoded location. The lookup data set contains the name of the county or parish.

The attribute variables STATENAME and COUNTYNM are missing where the value for _MATCHED_ is "None." or "City". The "City" observations were matched with multiple city-and-state observations in the lookup data set. As a result, the correct values for the attribute variables cannot be determined.

Output

The following output from the PRINT procedure shows the GEOCODED_CUSTOMERS output data set after running the GEOCODE procedure

Output 5.2 *The GEOCODED_CUSTOMERS Output Data Set with Additional Variables*

X	Y	_MATCHED_	STATENAME	COUNTYNM	address	zip
-118.399	34.0736	City			555 Junk Street	99999
-75.532	39.1500	ZIP	Delaware	Kent	115 E. Water St	19901
-75.570	39.0953	ZIP	Delaware	Kent	760 Moose Lodge Road	19934
-75.549	39.7366	ZIP	Delaware	New Castle	200 S. Madison Str	19801
-75.669	39.7317	ZIP	Delaware	New Castle	4701 Limestone Road	19808
-75.961	39.7877	ZIP	Pennsylvania	Chester	2117 N 4th St	19363
.	.	None			1313 Mockingbird Lane	.
-75.081	38.7265	ZIP	Delaware	Sussex	133 Silver Lake Dr	19971
-75.432	38.9035	ZIP	Delaware	Sussex	11 SE Front Street	19963
-75.611	38.6411	City			402 Nylon Boulevard	.
-75.605	39.2997	City			363 E Commerce St	.
-75.150	38.4663	ZIP	Delaware	Sussex	5595 Polly Branch Rd	19975
-75.053	38.4593	ZIP	Delaware	Sussex	1209 Coastal Highway	19944
-75.693	39.1509	ZIP	Delaware	Kent	2899 Arthursville Rd	19953
.	.	None			41 Bramhall St	.
-75.578	38.4564	City			9320 Old Racetrack Rd	.
-75.666	39.2282	ZIP	Delaware	Kent	281 W Commerce Str	19955
-75.850	39.6264	ZIP	Maryland	Cecil	211 Blue Ball Road	21921
-75.567	39.0695	ZIP	Delaware	Kent	3893 Turkey Point Rd	19980

Program

```
data CUSTOMERS (label="Customer data for geocoding");
infile datalines dlm='#';
length address $ 24 city $ 24 state $ 2;
input address    /* House number and street name */
      zip        /* Customer ZIP code (numeric)  */
      city       /* City name                    */
      state      /* State abbreviation           */
;
cust_ID = _n_;   /* Assign customer ID number    */
datalines;
555 Junk Street # 99999 # Beverly Hills # CA
115 E. Water St # 19901 # Dover #
760 Moose Lodge Road # 19934 # Camden #
200 S. Madison Str # 19801 # Wilmington # DE
4701 Limestone Road # 19808 # Wilmington #
2117 N 4th St # 19363 # Oxford # PA
1313 Mockingbird Lane # . # Delray # CC
133 Silver Lake Dr # 19971 # Rehoboth Beach # DE
11 SE Front Street # 19963 # Milford # DE
402 Nylon Boulevard # . # Seaford # DE
363 E Commerce St # . # Smyrna # DE
5595 Polly Branch Rd # 19975 # Selbyville # DE
1209 Coastal Highway # 19944 # Fenwick Island # DE
2899 Arthursville Rd # 19953 # Hartly # DE
41 Bramhall St # . #  #
9320 Old Racetrack Rd # . # Delmar # DE
281 W Commerce Str # 19955 # Kenton #
211 Blue Ball Road # 21921 # Elkton # MD
3893 Turkey Point Rd # 19980 # Woodside # DE
;
run;

proc geocode method=zip                          /* Geocoding method        */
             data=customers                      /* Address data            */
             out=geocoded_customers              /* Output data set         */
             attributevar=(statename, countynm); /* Include these variables */
run;

proc print data=geocoded_customers noobs;
  var x y _matched_ statename countynm address zip;
run;
```

Program Description

Generate the input data set of addresses that the GEOCODE procedure uses.

```
data CUSTOMERS (label="Customer data for geocoding");
infile datalines dlm='#';
length address $ 24 city $ 24 state $ 2;
input address    /* House number and street name */
      zip        /* Customer ZIP code (numeric)  */
      city       /* City name                    */
      state      /* State abbreviation           */
;
cust_ID = _n_;   /* Assign customer ID number    */
```

```
datalines;
555 Junk Street # 99999 # Beverly Hills # CA
115 E. Water St # 19901 # Dover #
760 Moose Lodge Road # 19934 # Camden #
200 S. Madison Str # 19801 # Wilmington # DE
4701 Limestone Road # 19808 # Wilmington #
2117 N 4th St # 19363 # Oxford # PA
1313 Mockingbird Lane # . # Delray # CC
133 Silver Lake Dr # 19971 # Rehoboth Beach # DE
11 SE Front Street # 19963 # Milford # DE
402 Nylon Boulevard # . # Seaford # DE
363 E Commerce St # . # Smyrna # DE
5595 Polly Branch Rd # 19975 # Selbyville # DE
1209 Coastal Highway # 19944 # Fenwick Island # DE
2899 Arthursville Rd # 19953 # Hartly # DE
41 Bramhall St # . #  #
9320 Old Racetrack Rd # . # Delmar # DE
281 W Commerce Str # 19955 # Kenton #
211 Blue Ball Road # 21921 # Elkton # MD
3893 Turkey Point Rd # 19980 # Woodside # DE
;
run;
```

Run the GEOCODE procedure with the generated input data set. Include in the output the state name and county name variables.

```
proc geocode method=zip                        /* Geocoding method        */
             data=customers                    /* Address data            */
             out=geocoded_customers            /* Output data set         */
             attributevar=(statename, countynm); /* Include these variables */
run;
```

Print the specified variable values from the GEOCODED_CUSTOMERS output data set, suppressing the observation column.

```
proc print data=geocoded_customers noobs;
  var x y _matched_ statename countynm address zip;
run;
```

Example 3: Street Geocoding

Features: STREET geocoding method

Procedure Options
- METHOD=
- DATA=
- OUT=
- LOOKUPSTREET=
- TYPE=

Other features: Base SAS functions
- SAS DATA step
- PRINT procedure

Data set: SASHELP.GEOEXM (primary street lookup data set)

Sample library member: GEOSTRT

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

This example illustrates the STREET geocoding method to obtain coordinates based on street addresses with house numbers.

Output

The following output from the PRINT procedure shows the default output variables in the WORK.GEOCODED output data set after running the GEOCODE procedure. For detailed explanations of the token values in the _NOTES_ column, see "Street Geocoding Note Values" on page 102.

Output 5.3 *The WORK.GEOCODED Data Set with STREET Method Output Variables*

address	M_ADDR	M_ZIP	M_OBS	_MATCHED_	_STATUS_	_NOTES_	_SCORE_	X	Y
555 Junk Street		.	12550	City	City/State Match	CT ST	10	-118.399	34.0736
305 Cross Lake Drive	305 Cross Lake Dr	27526	4863	Street	Found	AD ZC NM TS	65	-78.763	35.6060
2525 Banks Road	2525 Banks Rd	27603	1201	Street	Found	AD ZC NM TS	65	-78.673	35.6369
2222 SAS Campus Drive	199 Sas Campus Dr	27513	16146	Street	Found	AD ZC ENDNM TS	55	-78.763	35.8273
1150 SE Maynard Rd.	1150 SE Maynard Rd	27511	11729	Street	Found	AD ZC NM DP TS	80	-78.764	35.7831
2117 Graceland	4400 Graceland Ct	27606	7645	Street	Found	AD ZC ENDNM NOTSA	25	-78.711	35.7889
1313 Mockingbird Lane		.	.	None			0	.	.
133 Jade Circle	133 Jade Cir	27545	9404	Street	Found	AD ZC NM TS	65	-78.461	35.8144
1005 W South St	1005 W South St	27603	16956	Street	Found	AD ZC NM DP TS	80	-78.654	35.7732
622 Roundabout Road	399 Round About Rd	27540	15755	Street	Found	AD ZC ENDNM TS	55	-78.828	35.6497
Johnson Family Rd		27526	10797	ZIP	ZIP match	ZC	15	-78.784	35.5985
822 Water Plant Road	822 Water Plant Rd	27597	19482	Street	Found	AD CT ST NM TS	60	-78.340	35.8311
502 Possum Track Road	11800 Possum Track Rd	27614	14621	Street	Found	AD ZC ENDNM TS	55	-78.638	35.9477
2590 Wolfpack Lane	Wolfpack Ln	27604	20341	Street	Found	AD ZC NOLNM TS	55	-78.609	35.8235
125 Ferris Wheel Ct	125 Ferris Wheel Ct	27513	6650	Street	Found	AD ZC NM TS	65	-78.800	35.7949
3900 Western Blv	3900 Western Blvd	27606	19695	Street	Found	AD ZC NM TS	65	-78.691	35.7845

Program

```
data WORK.CUSTOMERS (label='Input data for street geocoding');
  infile datalines dlm='#';
    length address $ 32
           city    $ 24
           state   $ 2;
   input address  /* House number and street name      */
         zip      /* Customer ZIP code (numeric)       */
         city     /* City name                         */
```

```
         state;   /* Two-character postal abbreviation */
datalines;
555 Junk Street # 99999 # Beverly Hills # CA
305 Cross Lake Drive # 27526 # Fuquay-Varina # NC
2525 Banks Road # 27603 # Raleigh # NC
2222 SAS Campus Drive # 27513 # Cary # NC
1150 SE Maynard Rd. # 27511 # Cary # NC
2117 Graceland # 27606 # Raleigh # NC
1313 Mockingbird Lane # # Delray # CC
133 Jade Circle # 27545 # Knightdale # NC
1005 W South St # 27603 # Raleigh # NC
622 Roundabout Road # 27540 # Holly Springs # NC
Johnson Family Rd # 27526 # #
822 Water Plant Road # # Zebulon # NC
502 Possum Track Road # 27614 # # NC
2590 Wolfpack Lane # 27604 # Raleigh # NC
125 Ferris Wheel Ct # 27513 # Cary # NC
3900 Western Blv # 27606 # Raleigh # NC;
run;

/* Create custom street type data set which includes the non-standard
   abbreviation for 'BOULEVARD' used in the input data. */
data work.gctype;
  set sashelp.gctype;
  output;
  /* Insert non-standard type. */
  if _n_ = 1 then do;
    name  = 'BLV';
    type  = 'BLVD';
    group = 34; /* group number must match other 'BLVD' types */
    output;
  end;
run;
/* Sort the custom gctype dataset */
proc sort data=work.gctype;
  by name;
run;

proc geocode                          /* Invoke geocoding procedure        */
   method=STREET                      /* Specify geocoding method          */
   data=WORK.CUSTOMERS                /* Input data set of addresses       */
   out=WORK.GEOCODED                  /* Output data set with X/Y values   */
   lookupstreet=SASHELP.GEOEXM        /* Primary street lookup data set    */
   type=WORK.GCTYPE;                  /* Lookup data set-added street type */run;

proc print data=WORK.GEOCODED noobs;
   var address m_addr m_zip m_obs _matched_ _status_  _notes_ _score_ x y;
run;
```

Program Description

Generate the WORK_CUSTOMERS input data set of addresses that the GEOCODE procedure uses.

```
data WORK.CUSTOMERS (label='Input data for street geocoding');
  infile datalines dlm='#';
    length address $ 32
```

```
          city    $ 24
          state   $ 2;
   input address  /* House number and street name      */
         zip      /* Customer ZIP code (numeric)       */
         city     /* City name                         */
         state;   /* Two-character postal abbreviation */
datalines;
555 Junk Street # 99999 # Beverly Hills # CA
305 Cross Lake Drive # 27526 # Fuquay-Varina # NC
2525 Banks Road # 27603 # Raleigh # NC
2222 SAS Campus Drive # 27513 # Cary # NC
1150 SE Maynard Rd. # 27511 # Cary # NC
2117 Graceland # 27606 # Raleigh # NC
1313 Mockingbird Lane # # Delray # CC
133 Jade Circle # 27545 # Knightdale # NC
1005 W South St # 27603 # Raleigh # NC
622 Roundabout Road # 27540 # Holly Springs # NC
Johnson Family Rd # 27526 # #
822 Water Plant Road # # Zebulon # NC
502 Possum Track Road # 27614 # # NC
2590 Wolfpack Lane # 27604 # Raleigh # NC
125 Ferris Wheel Ct # 27513 # Cary # NC
3900 Western Blv # 27606 # Raleigh # NC;
run;
```

Create a custom street type lookup data set that includes the nonstandard abbreviation (Blv) for Boulevard used in the input data.

```
/* Create custom street type data set which includes the non-standard
   abbreviation for 'BOULEVARD' used in the input data. */
data work.gctype;
  set sashelp.gctype;
  output;
  /* Insert non-standard type. */
  if _n_ = 1 then do;
    name  = 'BLV';
    type  = 'BLVD';
    group = 34; /* group number must match other 'BLVD' types */
    output;
  end;
run;
/* Sort the custom gctype dataset */
proc sort data=work.gctype;
  by name;
run;
```

Run the GEOCODE procedure with the generated input data set.

```
proc geocode                          /* Invoke geocoding procedure        */
   method=STREET                      /* Specify geocoding method          */
   data=WORK.CUSTOMERS                /* Input data set of addresses       */
   out=WORK.GEOCODED                  /* Output data set with X/Y values   */
   lookupstreet=SASHELP.GEOEXM        /* Primary street lookup data set    */
   type=WORK.GCTYPE;                  /* Lookup data set-added street type */run;
```

Print the specified variable values from the WORK_GEOCODED output data set, suppressing the observation column.

```
proc print data=WORK.GEOCODED noobs;
   var address m_addr m_zip m_obs _matched_ _status_  _notes_ _score_ x y;
run;
```

Example 4: U.S. City Geocoding

Features: CITY geocoding method

Procedure Options
- METHOD=
- DATA=
- OUT=

Other features: Base SAS functions
- SAS DATA step
- PRINT procedure

Data set: mapsgfk.uscity_all (lookup data set)

Sample library member: GEOCITY

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

This example illustrates the CITY geocoding method to obtain coordinates based on a United States city and state.

There is no need to use the ADDRESSCOUNTRYVAR= option because all of the cities to be geocoded are located in the United States. The LOOKUPCITY= option is absent, which indicates that no alternate lookup data set is specified. In these circumstances, the GEOCODE procedure uses the default MAPSGFK.USCITY_ALL lookup data set for CITY geocoding within the United States.

Output

The following output from the PRINT procedure shows the output data set after running the GEOCODE procedure.

The output data set contains the LAT and LONG variables from the default lookup data set (MAPSGFK.USCITY_ALL).

Output 5.4 *The GEOCODED_TRAINING Output Data Set*

LAT	LONG	M_OBS	_MATCHED_	city	state
33.7489	-84.388	26936	City	Atlanta	GA
40.6806	-74.646	81289	City	Bedminster	NJ
35.7914	-78.781	93762	City	Cary	NC
41.8500	-87.650	32440	City	Chicago	IL
39.7589	-84.192	98332	City	Dayton	OH
41.6006	-93.609	40883	City	Des Moines	IA
42.3314	-83.046	65729	City	Detroit	MI
41.7636	-72.686	18459	City	Hartford	CT
25.7739	-80.194	22855	City	Miami	FL
42.5950	-71.017	61375	City	Middleton	MA
44.9800	-93.264	66546	City	Minneapolis	MN
40.7142	-74.006	85948	City	New York	NY
38.9822	-94.671	42171	City	Overland Park	KS
39.9522	-75.164	110328	City	Philadelphia	PA
33.4483	-112.073	6610	City	Phoenix	AZ
29.4239	-98.493	123723	City	San Antonio	TX
37.7750	-122.418	14956	City	San Francisco	CA
47.6064	-122.331	142724	City	Seattle	WA

Program

```
data training (label='Selected SAS training locations in USA');
  infile datalines dlm=',';
  length city  $ 24
         state $ 2;
  input city   /* City name                          */
        state; /* Two-character postal abbreviation */
datalines;
Atlanta, GA
Bedminster, NJ
Cary, NC
Chicago, IL
Dayton, OH
Des Moines, IA
Detroit, MI
Hartford, CT
Miami, FL
Middleton, MA
Minneapolis, MN
New York, NY
Overland Park, KS
Philadelphia, PA
Phoenix, AZ
San Antonio, TX
San Francisco, CA
Seattle, WA
```

```
;
run;

proc geocode
  method=city               /* Specify geocoding method       */
  data=training             /* Input data set of cities       */
  out=geocoded_training;    /* Output data set with locations */
run;

proc print data=geocoded_training noobs;
run;
```

Program Description

Generate the TRAINING input data set that the GEOCODE procedure uses.

```
data training (label='Selected SAS training locations in USA');
  infile datalines dlm=',';
  length city  $ 24
         state $ 2;
  input city   /* City name                          */
        state; /* Two-character postal abbreviation */
datalines;
Atlanta, GA
Bedminster, NJ
Cary, NC
Chicago, IL
Dayton, OH
Des Moines, IA
Detroit, MI
Hartford, CT
Miami, FL
Middleton, MA
Minneapolis, MN
New York, NY
Overland Park, KS
Philadelphia, PA
Phoenix, AZ
San Antonio, TX
San Francisco, CA
Seattle, WA
;
run;
```

Run the GEOCODE procedure with the generated TRAINING input data set. Specify the CITY method to add the latitude and longitude geographic coordinate values from the lookup data set to the addresses in the input data set. Those values are output to the GEOCODED_TRAINING data set. The default lookup data set is MAPSGFK.USCITY_ALL.

```
proc geocode
  method=city               /* Specify geocoding method       */
  data=training             /* Input data set of cities       */
  out=geocoded_training;    /* Output data set with locations */
run;
```

Print the entire GEOCODED_TRAINING output data set, suppressing the observation column.

```
proc print data=geocoded_training noobs;
run;
```

Example 5: World City Geocoding

Features: CITY geocoding method

Procedure options
- METHOD=
- DATA=
- OUT=
- ADDRESSCITYVAR=
- ADDRESSCOUNTRYVAR=
- ADDRESSSTATEVAR=PROVINCE
- ATTRIBUTEVAR=

Other features: Base SAS functions
- SAS DATA step
- PRINT procedure

Data set: MAPSGFK.WORLD_CITIES.

Sample library member: GEOWCITY

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

This example illustrates the CITY geocoding method to obtain coordinates of various international cities.

Output

The following output from the PRINT procedure shows the output data set after running the GEOCODE procedure.

The first two observations of the same city (Bella Vista, Argentina) illustrate how the variable named ADDRESSSTATEVAR is used. Note in the GEOCODED_CITIES output that the LAT and LONG values for the first Bella Vista observation are missing. There are two cities named Bella Vista in Argentina, but they are in different provinces. The first observation in the input data has a missing value for the PROVINCE variable, so the GEOCODE procedure cannot determine which of the matching cities to select. The _MATCHED_ value of '2 cities' shows the number of cities that were found. The second Bella Vista observation has a definite match and contains valid LAT and LONG coordinate variable values from the lookup data set. The GEOCODE procedure uses the province name 'Corrientes' in the input data to determine which of the two matching cities in the lookup data set was wanted.

Output 5.5 *The GEOCODED_CITIES Output Data Set with CITY Method Variables*

LAT	LONG	ISONAME	MapIDName1	M_OBS	_MATCHED_	city	province	countryID
.	.			.	2 cities	Bella Vista		ARG
-28.5167	-59.033	Argentina	Corrientes	496	City	Bella Vista	Corrientes	ARG
-34.9278	138.600	Australia	South Australia	920	City	Adelaide		AUS
46.8325	-71.237	Canada		5686	City	Quebec	Quebec	CAN
31.2382	121.469	China	Shanghai Shi	83459	City	Shanghai		CHN
56.0283	9.927	Denmark	Region Midtjylland	132492	City	Skanderborg		DNK
41.4049	2.187	Spain	Cataluna	132852	City	Barcelona		ESP
59.4206	24.709	Estonia	Harju maakond	132720	City	Tallinn		EST
55.8531	-4.218	United Kingdom	Glasgow City	134873	City	Glasgow		GBR
18.5154	73.859	India	MAHARASHTRA	139631	City	Pune		IND
53.3490	-6.271	Ireland	Leinster	137102	City	Dublin		IRL
37.5351	126.981	Korea, Republic of	Seoul	155379	City	Seoul		KOR
49.6114	6.132	Luxembourg	Luxembourg-ville	162109	City	Luxembourg		LUX
54.6887	25.290	Lithuania	Vilniaus	162096	City	Vilnius		LTU
25.6622	-100.406	Mexico	Nuevo Leon	163049	City	Garza Garcia		MEX
48.1543	17.126	Slovakia	Bratislavsky	194458	City	Bratislava		SVK
59.3248	18.073	Sweden	Stockholms	194375	City	Stockholm		SWE
13.7308	100.521	Thailand	Bangkok	195159	City	Bangkok		THA
29.7631	-95.363	United States	Texas	212746	City	Houston	Texas	USA
-26.2068	28.038	South Africa		215380	City	Johannesburg		ZAF

Program

```
data cities (label='International cities');
  infile datalines dlm=',';
  length city $32 province $24 countryID $3;
  input city        /* City name                    */
        province    /* State/province/district name */
        countryID;  /* Three-character country code */
datalines;
Bella Vista, , ARG
Bella Vista, Corrientes, ARG
Adelaide, , AUS
Quebec, Quebec, CAN
Shanghai, , CHN
Skanderborg, , DNK
Barcelona, , ESP
Tallinn, , EST
Glasgow, , GBR
Pune, , IND
Dublin, , IRL
Seoul, , KOR
Luxembourg, , LUX
Vilnius, , LTU
Garza Garcia, , MEX
Bratislava, , SVK
Stockholm, , SWE
Bangkok, , THA
Houston, Texas, USA
```

```
Johannesburg, , ZAF
;
run;

proc geocode                          /* Invoke geocoding procedure           */
   method=city                        /* Specify geocoding method             */
   data=cities                        /* Input data set of cities             */
   out=geocoded_cities                /* Output data set of locations         */
   addresscityvar=city                /* City name                            */
   addresscountryvar=countryID        /* Required for international geocoding  */
   addressstatevar=province           /* Optional state/province/district name */
   attributevar=(isoname mapidname1); /* Values to assign to geocoded cities   */
run;

proc print data=geocoded_cities noobs;
run;
```

Program Description

This example produces an output data set listing coordinates for various international cities. Because the LOOKUPCITY= data set option is omitted, the GEOCODE procedure uses the default GfK GeoMarketing world lookup data set MAPSGFK.WORLD_CITIES. This data set is an abridged version of world cities. If the more complete version (MAPSGFK.WORLD_CITIES_ALL) is downloaded from SAS MapsOnline (http://support.sas.com/rnd/datavisualization/mapsonline/html/downloads.html), use the LOOKUPCITY= option in PROC GEOCODE to reference it. The ADDRESSSTATEVAR=PROVINCE option is used to specify the input data set variable containing a city's corresponding state, province, or district value.

Generate the CITIES input data set that the GEOCODE procedure uses.

```
data cities (label='International cities');
  infile datalines dlm=',';
  length city $32 province $24 countryID $3;
  input city        /* City name                    */
        province    /* State/province/district name */
        countryID;  /* Three-character country code */
datalines;
Bella Vista, , ARG
Bella Vista, Corrientes, ARG
Adelaide, , AUS
Quebec, Quebec, CAN
Shanghai, , CHN
Skanderborg, , DNK
Barcelona, , ESP
Tallinn, , EST
Glasgow, , GBR
Pune, , IND
Dublin, , IRL
Seoul, , KOR
Luxembourg, , LUX
Vilnius, , LTU
Garza Garcia, , MEX
Bratislava, , SVK
Stockholm, , SWE
Bangkok, , THA
Houston, Texas, USA
```

```
Johannesburg, , ZAF
;
run;
```

Run the GEOCODE procedure with the generated CITIES input data set. Specify that GEOCODE use the city method. When geocoding international cities, you are required to use the ADDRESSCOUNTRYVAR= option to indicate that the cities are not all within the United States. Because the LOOKUPCITY= data set option is omitted, the GEOCODE procedure uses the default world lookup data set MAPSGFK.WORLD_CITIES. The procedure produces the GEOCODED_CITIES output data set. The output data set contains the LAT and LONG coordinate variables from the lookup data set.

```
proc geocode                          /* Invoke geocoding procedure           */
   method=city                        /* Specify geocoding method             */
   data=cities                        /* Input data set of cities             */
   out=geocoded_cities                /* Output data set of locations         */
   addresscityvar=city                /* City name                            */
   addresscountryvar=countryID        /* Required for international geocoding */
   addressstatevar=province           /* Optional state/province/district name */
   attributevar=(isoname mapidname1); /* Values to assign to geocoded cities  */
run;
```

Print the entire GEOCODED_CITIES output data set, suppressing the observation column.

```
proc print data=geocoded_cities noobs;
run;
```

Example 6: ZIP+4 Geocoding

Features: PLUS4 geocoding method

Procedure Options
- METHOD=
- LOOKUP=
- DATA=
- OUT=

Other features: Base SAS functions
- SAS DATA step
- PRINT procedure

Data set: lookup.zip4 (from SAS MapsOnline)

Sample library member: GEOZIP4

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

This example illustrates the PLUS4 geocoding method using US Postal Service ZIP and ZIP+4 postal codes. Lookup data can be downloaded from SAS MapsOnline (http://support.sas.com/rnd/datavisualization/mapsonline/html/downloads.html). You can also

purchase from Melissa Data the GEO*Data product containing ZIP+4 centroids (http://www.melissadata.com/reference-data/geodata.htm).

Output

The following output from the PRINT procedure shows the GEOCODED_CUSTOMERS output data set after running the GEOCODE procedure.

The example output shows four results for the customer names and addresses submitted as input. These include the location coordinates (X, Y), the matching observation's (row) number from the lookup data set (M_OBS), and the method with which GEOCODE made the match.

The _MATCHED_ column value in the output data set indicates how each address was located:

ZIP+4 - The ZIP and ZIP+4 values were matched. The location is the center of the street segment.

ZIP mean - The ZIP+4 value was not matched and the GEOCODE procedure used the ZIP method where multiple ZIP code matches were averaged.

ZIP - The ZIP+4 value was not matched and the GEOCODE procedure used the ZIP method where a single ZIP code match was found.

City - Neither the ZIP+4 nor the ZIP code were found but the CITY method returned a match.

Output 5.6 *The GEOCODED_CUSTOMERS Output Data Set with PLUS4 Method Variables*

Y	X	M_OBS	_MATCHED_	name	city	state	zip	plus4
35.7472	-78.6747	3615346	ZIP+4	J. Cheever Loophole	Raleigh	NC	27603	2681
29.7709	-95.7551	13324415	ZIP+4	Cuthbert J. Twillie	Katy	TX	77450	3418
39.7424	-75.6931	2556375	ZIP+4	Kaspar Gutman	Wilmington	DE	19808	1927
39.7809	-75.9782	2482823	ZIP+4	Dr. Hugo Z. Hackenbush	Oxford	PA	19363	1735
35.1728	-78.0432	.	ZIP mean	Charlie Allnut	Mount Olive	NC	28365	2277
38.9124	-75.4265	2574841	ZIP+4	Larson E. Whipsnade	Milford	DE	19963	1941
27.4986	-82.5750	22587	City	Guillermo Ugarte	Bradenton	FL	.	2208
38.4644	-75.2211	2579981	ZIP+4	Capt. Geoffrey Spaulding	Selbyville	DE	19975	7504
38.4619	-75.0534	.	ZIP mean	Joel Cairo	Fenwick Island	DE	19944	4401
39.1706	-75.7132	2571896	ZIP+4	Charles Blutoski	Hartly	DE	19953	3141
29.9972	-98.0983	126544	City	Rufus T. Firefly	Wimberley	TX	.	.
39.2282	-75.6658	2572140	ZIP	Otis B. Driftwood	Kenton		19955	53
39.6120	-75.8452	2888589	ZIP+4	Gordon Miller	Elkton	MD	21921	5335

Program

```
libname lookup 'pathname';

data customers;
   infile datalines dlm=',';
   length name city $32 state $2;
   input name      /* Customer name      */
         zip       /* Customer ZIP code  */
         plus4     /* ZIP+4              */
         city      /* City name          */
         state;    /* State abbreviation */
datalines;
```

```
J. Cheever Loophole, 27603, 2681, Raleigh, NC
Cuthbert J. Twillie, 77450, 3418, Katy, TX
Kaspar Gutman, 19808, 1927, Wilmington, DE
Dr. Hugo Z. Hackenbush, 19363, 1735, Oxford, PA
Charlie Allnut, 28365, 2277, Mount Olive, NC
Larson E. Whipsnade, 19963, 1941, Milford, DE
Guillermo Ugarte, ., 2208, Bradenton, FL
Capt. Geoffrey Spaulding, 19975, 7504, Selbyville, DE
Joel Cairo, 19944, 4401, Fenwick Island, DE
Charles Blutoski, 19953, 3141, Hartly, DE
Rufus T. Firefly, ., ., Wimberley, TX
Otis B. Driftwood, 19955, 53, Kenton, ,
Gordon Miller, 21921, 5335, Elkton, MD
;
run;

proc geocode           /* Invoke geocoding procedure  */
     method=plus4             /* Specify geocoding method    */
     lookup=lookup.zip4       /* Lookup data from MapsOnline */
     data=customers           /* Input data set to geocode   */
     out=geocoded_customers;  /* Specify name of Output data set of locations */
run;

proc print data=geocoded_customers noobs;
run;
```

Program Description

The LIBNAME statement assigns the library name LOOKUP to the location where the ZIP+4 lookup data is installed. You must edit the 'pathname' in the following LIBNAME statement to reference the lookup data location on your system.

```
libname lookup 'pathname';
```

Generate the CUSTOMERS input data set of names and addresses for the GEOCODE procedure to use.

```
data customers;
   infile datalines dlm=',';
   length name city $32 state $2;
   input name      /* Customer name      */
         zip       /* Customer ZIP code  */
         plus4     /* ZIP+4              */
         city      /* City name          */
         state;    /* State abbreviation */
datalines;
J. Cheever Loophole, 27603, 2681, Raleigh, NC
Cuthbert J. Twillie, 77450, 3418, Katy, TX
Kaspar Gutman, 19808, 1927, Wilmington, DE
Dr. Hugo Z. Hackenbush, 19363, 1735, Oxford, PA
Charlie Allnut, 28365, 2277, Mount Olive, NC
Larson E. Whipsnade, 19963, 1941, Milford, DE
Guillermo Ugarte, ., 2208, Bradenton, FL
Capt. Geoffrey Spaulding, 19975, 7504, Selbyville, DE
Joel Cairo, 19944, 4401, Fenwick Island, DE
Charles Blutoski, 19953, 3141, Hartly, DE
Rufus T. Firefly, ., ., Wimberley, TX
```

```
Otis B. Driftwood, 19955, 53, Kenton, ,
Gordon Miller, 21921, 5335, Elkton, MD
;
run;
```

Run the GEOCODE procedure with the generated CUSTOMERS input data set.
Specify that GEOCODE use the PLUS4 method to look up each address.

```
proc geocode           /* Invoke geocoding procedure  */
     method=plus4             /* Specify geocoding method    */
     lookup=lookup.zip4       /* Lookup data from MapsOnline */
     data=customers           /* Input data set to geocode   */
     out=geocoded_customers;  /* Specify name of Output data set of locations */
run;
```

Print the entire GEOCODED_CUSTOMERS output data set, suppressing the observation column.

```
proc print data=geocoded_customers noobs;
run;
```

Example 7: British Postcode Geocoding

Features: ZIP geocoding method

Procedure Options
- METHOD=
- DATA=
- OUT=
- LOOKUP=
- ADDRESSZIPVAR=
- LOOKUPZIPVAR=
- LOOKUPLONGVAR=
- LOOKUPLATVAR=
- ATTRIBUTEVAR=

Other features: Base SAS functions
- SAS DATA step
- PRINT procedure

Data set: lookup.postcodes (Postcode lookup data from MapsOnline)

Sample library member: GEOZIPUK

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

This example illustrates the ZIP geocoding method using British Royal Mail postcodes. Free lookup data can be downloaded from the British Ordnance Survey. Use the %CODEPOINT2GEOCODE SAS macro program to import the British postcode data. You can download this macro program from SAS MapsOnline (http://support.sas.com/rnd/datavisualization/mapsonline/html/downloads.html). Instructions for downloading and importing the British Royal Mail postcodes are included with the

%CODEPOINT2GEOCODE macro program. See "Non-U.S. Postcodes" on page 118 for details.

Output

The following output from the PRINT procedure shows the GEOCODED_OFFICES output data set after running the GEOCODE procedure.

Output 5.7 *The GEOCODED_OFFICES Output Data Set with ZIP Method Variables*

Y	X	Y_DMS	X_DMS	M_OBS	_MATCHED_	name	city	postcode	country
53.4702	-2.29462	53° 28' 12.59"	-2° 17' 40.65"	877702	ZIP	Quay Plaza	Manchester	M503BA	UK
51.5573	-0.81686	51° 33' 26.39"	00° 49' 00.68"	1356611	ZIP	Wittington House	Buckinghamshire	SL72EB	UK
55.8639	-4.25521	55° 51' 49.93"	-4° 15' 18.77"	535160	ZIP	Tara House	Glasgow	G21HG	UK
51.5172	-0.08385	51° 31' 01.76"	00° 05' 01.87"	460699	ZIP	New Broad Street	London	EC2M1NH	UK

Program

```
libname lookup 'pathname';    /* Location of British postcode lookup data */

data offices;
  infile datalines dlm=',';
  length name city $24 postcode $8 country $2;
  input name                          /* Office name              */
        city                          /* Name of city             */
        postcode                      /* Royal Mail postcode      */
        country;                      /* Two-character country ID */
  postcode=upcase(compress(postcode)); /* Normalize postcodes      */
datalines;
Quay Plaza, Manchester, M50 3BA, UK
Wittington House, Buckinghamshire, SL7 2EB, UK
Tara House, Glasgow, G2 1HG, UK
New Broad Street, London, EC2M 1NH, UK
;
run;

proc geocode                      /* Invoke geocoding procedure            */
   method=zip                     /* Specify geocoding method              */
   data=work.offices              /* Input data set of offices             */
   out=geocoded_offices           /* Output data set of locations          */
   lookup=lookup.postcodes        /* Postcode lookup data from MapsOnline  */
   addresszipvar=postcode         /* Postcode variable in input data       */
   lookupzipvar=pc                /* Postcode variable in lookup data      */
   attributevar=(Y_dms            /* Additional variables from lookup data */
                 X_dms);          /* set to assign to geocoded locations   */
run;

proc print data=geocoded_offices noobs;
run;
```

Program Description

The LIBNAME statement assigns the library name LOOKUP to the location where the postcodes lookup data set is installed. This is the lookup data imported from the Ordnance Survey by the %CODEPOINT2GEOCODE macro program. You must

edit the 'pathname' in the following LIBNAME statement to reference the lookup data location on your system.

```
libname lookup 'pathname';    /* Location of British postcode lookup data */
```

Generate the OFFICES input data set of addresses that the GEOCODE procedure uses. Each variable is delimited by a comma. Maximum variable lengths are specified. An important step is to remove spaces and capitalize the postcode values.

```
data offices;
  infile datalines dlm=',';
  length name city $24 postcode $8 country $2;
  input name                          /* Office name              */
        city                          /* Name of city             */
        postcode                      /* Royal Mail postcode      */
        country;                      /* Two-character country ID */
  postcode=upcase(compress(postcode)); /* Normalize postcodes      */
datalines;
Quay Plaza, Manchester, M50 3BA, UK
Wittington House, Buckinghamshire, SL7 2EB, UK
Tara House, Glasgow, G2 1HG, UK
New Broad Street, London, EC2M 1NH, UK
;
run;
```

Run the GEOCODE procedure with the generated OFFICES input data set located in the WORK folder. The GEOCODE procedure uses the postcode from each address in the input data to match observations in the lookup data set.

```
proc geocode                     /* Invoke geocoding procedure           */
   method=zip                    /* Specify geocoding method             */
   data=work.offices             /* Input data set of offices            */
   out=geocoded_offices          /* Output data set of locations         */
   lookup=lookup.postcodes       /* Postcode lookup data from MapsOnline  */
   addresszipvar=postcode        /* Postcode variable in input data      */
   lookupzipvar=pc               /* Postcode variable in lookup data     */
   attributevar=(Y_dms           /* Additional variables from lookup data */
                 X_dms);         /* set to assign to geocoded locations  */
run;
```

Print the entire GEOCODED_OFFICES output data set, suppressing the observation column.

```
proc print data=geocoded_offices noobs;
run;
```

Example 8: Australian Postcode Geocoding

Features: ZIP geocoding method

Procedure Options

- METHOD=
- DATA=
- OUT=
- LOOKUP=
- ADDRESSZIPVAR=

LOOKUPZIPVAR=

Other features: Base SAS functions
SAS DATA step
PRINT procedure

Data set: lookup.postcodes (Postcode lookup data from MapsOnline)

Sample library member: GEOZIPAU

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

This example illustrates the ZIP geocoding method using Australian postcodes. Free lookup data can be downloaded from the Australian Bureau of Statistics. Use the %ABS2GEOCODE SAS macro program to import the Australian postcode data. You can download this macro program from SAS MapsOnline (http://support.sas.com/rnd/datavisualization/mapsonline/html/downloads.html). Instructions for downloading and importing the Australian postcodes are included with the %ABS2GEOCODE macro program. See "Non-U.S. Postcodes" on page 118 for details.

Output

The following output from the PRINT procedure shows the GEOCODED_STADIUMS output data set after running the GEOCODE procedure.

Output 5.8 *The GEOCODED_STADIUIMS Output Data Set with ZIP Method Variables*

y	x	M_OBS	_MATCHED_	team	stadium	city	state	postcode
-33.8393	151.067	136	ZIP	South Sydney Rabbitohs	ANZ Stadium	Sydney	New South Wales	2127
-33.8915	151.229	43	ZIP	Sydney Roosters	Sydney Football Stadium	Moore Park	New South Wales	2021
-33.8393	151.067	136	ZIP	Canterbury Bulldogs	ANZ Stadium	Sydney	New South Wales	2127
-33.8152	151.008	158	ZIP	Parramatta Eels	Parramatta Stadium	Sydney	New South Wales	2150
-33.7462	151.256	111	ZIP	Manly-Warringah Sea Eagles	Brookvale Oval	Brookvale	New South Wales	2100
-34.0647	151.146	224	ZIP	Cronulla-Sutherland Sharks	Toyota Park	Woolooware	New South Wales	2230
-33.7508	150.679	557	ZIP	Penrith Panthers	Centrebet Stadium	Penrith	New South Wales	2750
-35.2284	149.092	475	ZIP	Canberra Raiders	Canberra Stadium	Bruce	Australian Capital Territory	2617
-27.4645	153.000	1360	ZIP	Brisbane Broncos	Suncorp Stadium	Milton	Queensland	4064
-32.9208	151.736	254	ZIP	Newcastle Knights	Hunter Stadium	New Castle	New South Wales	2292
-19.4336	146.212	1704	ZIP	North Queensland Cowboys	Dairy Farmers Stadium	Townsville	Queensland	4817
-37.8114	144.965	661	ZIP	Melbourne Storm	AAMI Park	Melbourne	Victoria	3000
-33.9717	151.141	211	ZIP	St. George Illawarra Dragons	WIN Jubilee Oval	Carlton	New South Wales	2217
-34.1461	150.802	433	ZIP	Wests Tigers	Campbelltown Stadium	Leumeah	New South Wales	2560
-33.8760	151.159	62	ZIP	Wests Tigers	Leichhardt Oval	Leichhardt	New South Wales	2040
-33.8915	151.229	43	ZIP	Wests Tigers	Sydney Football Stadium	Sydney	New South Wales	2021
-28.0626	153.389	1449	ZIP	Gold Coast Titans	Skilled Park	Gold Coast	Queensland	4226

Program

```
libname lookup 'pathname';   /* Location of Australian postcode lookup data */

data stadiums (label='Australian National Rugby League (NRL) stadiums');
  infile datalines dlm=',';
  length team $32 stadium $32 city $ 24 state $32 postcode $4;
  input team        /* NRL team name    */
        stadium     /* Stadium name     */
```

```
        city        /* City name        */
        state       /* State name       */
        postcode;   /* Stadium postcode */
datalines;
South Sydney Rabbitohs, ANZ Stadium, Sydney, New South Wales, 2127
Sydney Roosters, Sydney Football Stadium, Moore Park, New South Wales, 2021
Canterbury Bulldogs, ANZ Stadium, Sydney, New South Wales, 2127
Parramatta Eels, Parramatta Stadium, Sydney, New South Wales, 2150
Manly-Warringah Sea Eagles, Brookvale Oval, Brookvale, New South Wales, 2100
Cronulla-Sutherland Sharks, Toyota Park, Woolooware, New South Wales, 2230
Penrith Panthers, Centrebet Stadium, Penrith, New South Wales, 2750
Canberra Raiders, Canberra Stadium, Bruce, Australian Capital Territory, 2617
Brisbane Broncos, Suncorp Stadium, Milton, Queensland, 4064
Newcastle Knights, Hunter Stadium, New Castle, New South Wales, 2292
North Queensland Cowboys, Dairy Farmers Stadium, Townsville, Queensland, 4817
Melbourne Storm, AAMI Park, Melbourne, Victoria, 3000
St. George Illawarra Dragons, WIN Jubilee Oval, Carlton, New South Wales, 2217
Wests Tigers, Campbelltown Stadium, Leumeah, New South Wales, 2560
Wests Tigers, Leichhardt Oval, Leichhardt, New South Wales, 2040
Wests Tigers, Sydney Football Stadium, Sydney, New South Wales, 2021
Gold Coast Titans, Skilled Park, Gold Coast, Queensland, 4226
;
run;

proc geocode              /* Invoke geocoding procedure       */
  method=zip              /* Specify geocoding method         */
  data=stadiums           /* Input data of stadiums           */
  out=geocoded_stadiums   /* Output data set of locations     */
  lookup=lookup.postcodes /* Lookup data with postcodes       */
  addresszipvar=postcode  /* Postcode variable in input data  */
  lookupzipvar=poa_code;  /* Postcode variable in lookup data */
run;

proc print data=geocoded_stadiums noobs;
run;
```

Program Description

The LIBNAME statement assigns the library name LOOKUP to the location where the Australian postcode lookup data set is installed. This is the lookup data imported from the Australian Bureau of Statistics by the %ABS2GEOCODE macro program. You must edit the 'pathname' in the following LIBNAME statement to reference the lookup data location on your system.

```
libname lookup 'pathname';   /* Location of Australian postcode lookup data */
```

Generate the STADIUMS input data set of names and addresses that the GEOCODE procedure uses. Each variable is delimited by a comma. Maximum variable lengths are specified.

```
data stadiums (label='Australian National Rugby League (NRL) stadiums');
  infile datalines dlm=',';
  length team $32 stadium $32 city $ 24 state $32 postcode $4;
  input team        /* NRL team name    */
        stadium     /* Stadium name     */
        city        /* City name        */
        state       /* State name       */
```

```
        postcode;   /* Stadium postcode */
datalines;
South Sydney Rabbitohs, ANZ Stadium, Sydney, New South Wales, 2127
Sydney Roosters, Sydney Football Stadium, Moore Park, New South Wales, 2021
Canterbury Bulldogs, ANZ Stadium, Sydney, New South Wales, 2127
Parramatta Eels, Parramatta Stadium, Sydney, New South Wales, 2150
Manly-Warringah Sea Eagles, Brookvale Oval, Brookvale, New South Wales, 2100
Cronulla-Sutherland Sharks, Toyota Park, Woolooware, New South Wales, 2230
Penrith Panthers, Centrebet Stadium, Penrith, New South Wales, 2750
Canberra Raiders, Canberra Stadium, Bruce, Australian Capital Territory, 2617
Brisbane Broncos, Suncorp Stadium, Milton, Queensland, 4064
Newcastle Knights, Hunter Stadium, New Castle, New South Wales, 2292
North Queensland Cowboys, Dairy Farmers Stadium, Townsville, Queensland, 4817
Melbourne Storm, AAMI Park, Melbourne, Victoria, 3000
St. George Illawarra Dragons, WIN Jubilee Oval, Carlton, New South Wales, 2217
Wests Tigers, Campbelltown Stadium, Leumeah, New South Wales, 2560
Wests Tigers, Leichhardt Oval, Leichhardt, New South Wales, 2040
Wests Tigers, Sydney Football Stadium, Sydney, New South Wales, 2021
Gold Coast Titans, Skilled Park, Gold Coast, Queensland, 4226
;
run;
```

Run the GEOCODE procedure with the generated STADIUMS input data set. Specify that GEOCODE use the ZIP method to look up each stadium address. The input data contains address information about Australian National Rugby League stadiums. The GEOCODE procedure uses the postcode from each address in the input data to match observations in the lookup data set.

```
proc geocode              /* Invoke geocoding procedure       */
  method=zip              /* Specify geocoding method         */
  data=stadiums           /* Input data of stadiums           */
  out=geocoded_stadiums   /* Output data set of locations     */
  lookup=lookup.postcodes /* Lookup data with postcodes       */
  addresszipvar=postcode  /* Postcode variable in input data  */
  lookupzipvar=poa_code;  /* Postcode variable in lookup data */
run;
```

Print the entire GEOCODED_STADIUMS output data set, suppressing the observation column.

```
proc print data=geocoded_stadiums noobs;
run;
```

Chapter 6

GINSIDE Procedure

Overview: GINSIDE Procedure

The GINSIDE procedure compares a data set of X and Y coordinates to a map data set containing map polygons. The procedure determines whether the X and Y coordinates for each point fall inside or outside of the map polygons. If the point falls inside a polygon, then the ID variable is set to the ID value of that polygon. For example, if a map contains states, then the ID variable of the output data set is set to the state that contains the point. The GINSIDE procedure can be used with the SAS/GRAPH map data sets.

Note: Avoid unpredictable map results with points that fall on the border of a polygon by using the INCLUDEBORDER option. This option includes points that are on the border of a polygon in the output data set. These points are identified by the variable named _ONBORDER_ with a value of 1.

Syntax: GINSIDE Procedure

Requirement:

- One DATA= argument if no points data set was created before issuing the PROC GINSIDE statement
- One MAP= argument
- One ID statement

PROC GINSIDE
DATA=*points-data-set*
MAP=*map-data-set*
<option(s)>;
ID *id-variable(s)*;

PROC GINSIDE Statement

The GINSIDE procedure compares a data set of X and Y coordinates to a map data set containing map polygons. The procedure determines whether the X and Y points fall inside or outside of the map polygons.

Requirement: Three data sets are required: a data set containing points, a map data set, and an output data set.

Syntax

PROC GINSIDE
DATA=*points-data-set*
MAP=*map-data-set*
<OUT=*output-data-set* >
<DROPMAPVARS>
<INSIDEONLY>
<INCLUDEBORDER>
<KEEPMAPVARS>;

Summary of Optional Arguments

DROPMAPVARS
specifies that the GINSIDE procedure should keep the ID variable but drop all of the other map data set variables when writing to the output data set.

INCLUDEBORDER
includes points that are on the border of a polygon in the output data set.

INSIDEONLY
causes the output data set to contain only points that are inside the map polygons.

KEEPMAPVARS
specifies that the GINSIDE procedure should include all of the map data set variables in the output data set.

OUT=*output-data-set*
specifies the output data set for the GINSIDE procedure.

Required Arguments

DATA=*points-data-set*
specifies an input data set that contains the X and Y coordinates of the individual points that are being compared to the map polygons.

Default By default, the procedure uses the most recently created SAS data set.

Restriction This statement is required if no points data set was created before issuing the PROC GINSIDE statement.

Note If this data set contains the same ID variable (or variables) as does the map data set, the value should be set to MISSING. In this case the points are not considered to be part of the boundary of the polygon. Alternatively, you can rename the ID variable in the points data set, provided that it represents a different value than that of the ID variable in the map data set.

MAP=*map-data-set*
specifies the map data set that contains the polygons that you want to compare the points in the input data set to. This data must conform to the rules for a map data set and contain variables X and Y and one or more ID variables. The ID statement should name that variable or variables.

Restriction The X and Y values in the input data set must be in the same projection system and units as the X and Y in the map data set. If the map data set has unprojected X and Y values in radians, then the point data set X and Y variable values must also be unprojected and in radians.

Optional Arguments

DROPMAPVARS
specifies that the GINSIDE procedure should keep the ID variable but drop all of the other map data set variables when writing to the output data set. The GINSIDE procedure writes to the output data set all variables associated with each point whose X and Y coordinates fall inside the map polygon. Use the DROPMAPVARS option to circumvent this default behavior.

Alias DMV

Default The default behavior is to keep all map data set variables and this option overrides it.

See "KEEPMAPVARS" on page 177

INCLUDEBORDER
includes points that are on the border of a polygon in the output data set. If any points are on the border of a polygon, then the output data set includes a variable named _ONBORDER_. If the value of _ONBORDER_ is `1`, then the point is located on the border of a polygon.

Note: A point can be on the border between multiple polygons. In this case the point is assigned to the ID of the first polygon whose border it shares. This is the polygon that is processed first.

INSIDEONLY
causes the output data set to contain only points that are inside the map polygons. By default, the data set contains all points.

KEEPMAPVARS
specifies that the GINSIDE procedure should include all of the map data set variables in the output data set. The ID variable from the map data set is one of those included. By default, the GINSIDE procedure writes to the output data set all variables associated with each point whose X and Y coordinates fall inside the map polygon. If a point falls inside a polygon, then the ID variable is set to the ID value of that polygon. For example, if a map contains counties, then the ID variable is set to the ID value of that polygon for each data point found inside a county.

Alias	KMV
Default	The behavior specified by the KEEPMAPVARS option is the default. This option does not need to be specified.
Restriction	When comparing data points with map polygons, like-named variables must be of the same type or the GINSIDE procedure cannot run. For example, a LAKE character variable in a points data set cannot be compared with a numeric LAKE variable contained in a map data set.
See	"DROPMAPVARS" on page 177

OUT=*output-data-set*
specifies the output data set for the GINSIDE procedure.

By default, the output data set contains all of the observations and variables from the points data set, and all of the variables identifying the map area. This includes the ID variable from the map data set.

Default	If you do not specify a name for an output data set with the OUT= option then PROC GINSIDE creates one for you.
See	the option "DROPMAPVARS" on page 177 if you want the output data set to contain only the ID variable from the map data set.

ID Statement

Specifies the identification variables in the map data set whose polygons are checked against the points from the input data set.

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

Syntax

ID *id-variable(s)*;

Required Argument

id-variable(s)
specifies one or more identification variables from the map data set. For each X and Y point in the input data set and each ID variable that you specify, the procedure determines whether the point lies within a polygon in the map data set. If it does, then the ID value of that polygon is written to the output data set. If the point lies outside of all of the polygons, then a missing value is written to the output data set.

For example, consider an input data set that contains the values **`X=1.37`** and **`Y=.68`**. In the PROC GINSIDE statement, you specify the MAPSSAS.COUNTIES map data set. In the ID statement, you specify STATE and COUNTY as the lookup variables. The point (1.37,.68) lies within the polygon where STATE=54 and COUNTY=27. As a result, the value for STATE is 54 and for COUNTY is 27 in the corresponding observation in the output data set.

Optimizing Performance

Limiting the data that is input to PROC GINSIDE decreases the time it takes to run the procedure. This section offers several tips to help you enhance the performance of the GINSIDE procedure.

Limiting the Point Data and the Map Data

TIP Include only areas of interest.

For performance reasons, it is recommended, if possible, that you limit the point data and the map data to the known area of interest. For example, if you are interested in points that are only in the continental United States (U.S.), then limit your U.S. map to those 48 states. If your map data includes other areas, such as European data, then eliminate European areas from the map data.

Map Data Integrity

TIP Keep your data precise.

The results from the GINSIDE procedure are only as good as the precision of the map data that is input to the procedure. Imprecise map data, or map data that has been reduced, inputs to the procedure smooth borders rather than the original, jagged land edges. If a data point falls on a tiny piece of land that was smoothed out, then the GINSIDE procedure does not find the point inside the expected polygonal area. This can be especially noticeable with points along the coast.

Removing Duplicate Data Points

TIP Point data should not be repeated.

To increase efficiency, make sure that there is only one occurrence of a data point. If GINSIDE has to process the same data point multiple times, it slows down the performance of the procedure.

Examples: GINSIDE Procedure

Example 1: Determining Values By Using the GINSIDE Procedure

Features: MAP= required argument referring to map data set
DATA= required argument referring to data points
OUT= optional argument referring to output data set
ID statement

Other features: SAS DATA step with assignment statements
SORT procedure

PRINT procedure

Data sets: MAPSSAS.COUNTIES (map data)
GPSCOUNTIES (created data set)

Sample library member: GINSIDE2

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

This example uses the GINSIDE procedure to determine the state and county for each pair of coordinates in the input data set.

The following table shows the values of STATE and COUNTY for each observation in the input data set.

Output 6.1 *PROC PRINT Results of Output Data Set*

Obs	site	STATE	COUNTY	x	y
1	a	42	133	1.34451	0.69892
2	b	54	27	1.36910	0.68351
3	c	54	27	1.36854	0.68723
4	d	42	21	1.37470	0.70922

Program

```
goptions reset=global border;

data gpscounties;
  input longitude latitude site $;
  x=longitude*arcos(-1)/180;
  x=x*(-1);
  y=latitude*arcos(-1)/180;
datalines;
-77.0348 40.0454 a
-78.4437 39.1623 b
-78.4115 39.3751 c
-78.7646 40.6354 d
;
run;

proc ginside data=gpscounties map=mapssas.counties out=gpscounties;
  id state county;
run;

proc sort data=gpscounties;
  by site;
run;

proc print data=gpscounties;
  var site state county x y;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=global border;
```

Create the GPSCOUNTIES data set. The X and Y variables are converted from decimal degrees to radians. The X variable is also multiplied by –1 to switch the coordinates from east longitude to west longitude, in order to match the values in the MAPSSAS.COUNTIES data set.

```
data gpscounties;
  input longitude latitude site $;
  x=longitude*arcos(-1)/180;
  x=x*(-1);
  y=latitude*arcos(-1)/180;
datalines;
-77.0348 40.0454 a
-78.4437 39.1623 b
-78.4115 39.3751 c
-78.7646 40.6354 d
;
run;
```

Determine the values of STATE and COUNTY for each data point. Add the values to the created data set. Use the MAPSSAS libref to access the counties map data set.

```
proc ginside data=gpscounties map=mapssas.counties out=gpscounties;
  id state county;
run;
```

Sort and print the output data set.

```
proc sort data=gpscounties;
  by site;
run;

proc print data=gpscounties;
  var site state county x y;
run;
quit;
```

Example 2: Mapping and Annotating Values from the GINSIDE Procedure

Features: MAP= required argument referring to map data set
DATA= required argument referring to data points
OUT= optional argument referring to output data set
ID statement

Other features: SAS DATA step with assignment statements
GPROJECT procedure
PRINT procedure
GMAP procedure

Data sets: MAPSSAS.COUNTIES (map data)
MYMAP (created data set)
CUSTOMERS (created data set)

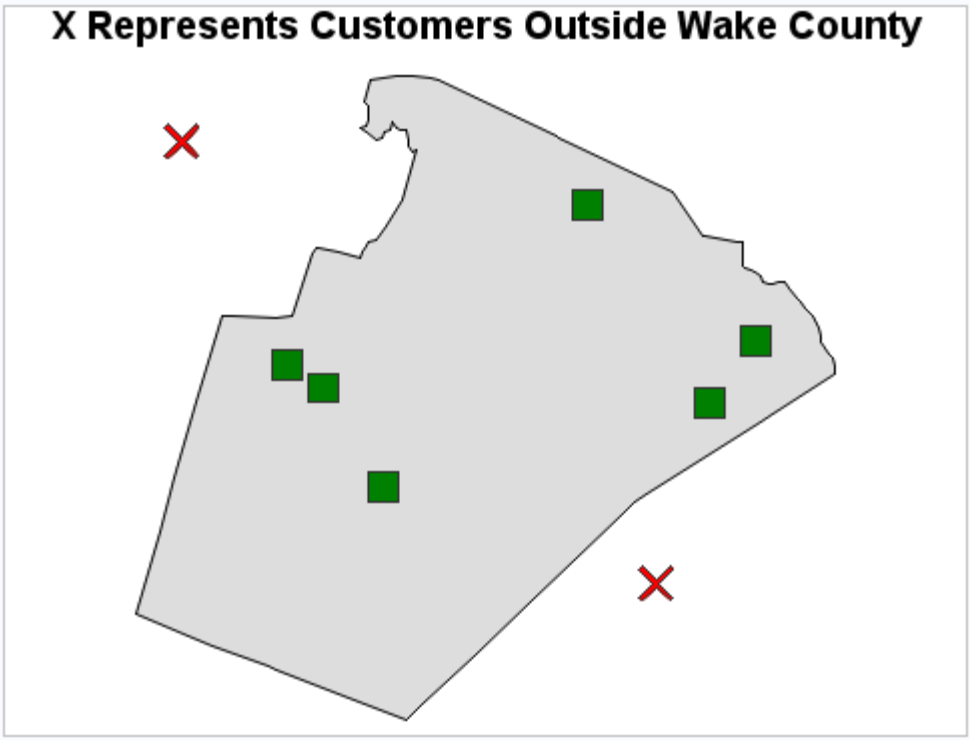
Sample library member: GINSIDE

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

The following example determines which customers are inside Wake County in the state of North Carolina. It then draws a map and colors the markers (representing customers) to distinguish customers inside the county from customers outside the county. It also displays the output data set. This example is featured in the SAS Sample Library under the name GINSIDE.

The map of Wake County displayed is the output annotated with customer data points both inside and outside of the county. The second graphic displays the CUSTOMERS data set, the result of running PROC PRINT. Notice that the last two observations in the printed table have missing values for COUNTY because they are not in Wake County.



Obs	x	y	SEGMENT	STATE	COUNTY	DENSITY	city	lastname	zip
1	-.001541863	-.001337842	.	37	183	.	Raleigh	Smith	27611
2	-.002986279	0.000506524	.	37	183	.	Morrisville	Jones	27560
3	-.002444484	0.000149461	.	37	183	.	Cary	Doe	27513
4	0.002555932	-.002802348	.	.	.	.	Clayton	Patel	27520
5	-.004565994	0.003861846	.	.	.	.	Durham	White	27705
6	0.001531260	0.002906157	.	37	183	.	WakeForest	Short	27587
7	0.003358788	-.000065627	.	37	183	.	Wendell	Phillips	27591
8	0.004053657	0.000860237	.	37	183	.	Zebulon	Jackson	27597

Program

```
goptions reset=global border;

data customers;
  length city $20;
  input lastname$ zip x y city $;

  datalines;
Smith    27611 1.374164 0.623436 Raleigh
Jones    27560 1.375948 0.625278 Morrisville
Doe      27513 1.375279 0.624922 Cary
Patel    27520 1.369120 0.621970 Clayton
White    27705 1.377910 0.628629 Durham
Short    27587 1.370373 0.627680 WakeForest
Phillips 27591 1.368124 0.624705 Wendell
Jackson  27597 1.367264 0.625629 Zebulon
;
run;

data mymap;
  set mapssas.counties(where=(fipstate(state)='NC' and county=183));
run;

data combined;
  set customers mymap;
run;

proc gproject data=combined out=combined dupok;
  id state county;
run;

data mymap customers;
  set combined;
  if missing(zip) = 1 then output mymap;
  else output customers;
run;

proc ginside map=mymap data=customers out=customers;
  id state county;
run;

proc print data=customers;
run;

data customers;
   set customers;
   length color style $8;
   retain xsys ysys '2' hsys '3' when 'a' position '5' size 5;
```

```
    style='marker';
    if missing(county) then do;
      color='red';
      text='X';
      end;
    else do;
      color='green';
      text='U';       /* Marker font for a square */
      end;
    output;
    style='markere';  /* outline the symbols */
    color='gray33';
    output;
  run;

title 'X Represents Customers Outside Wake County';
pattern1 v=s c=graydd;

proc gmap data=mymap map=mymap anno=customers;
  id state county;
  choro county / stat=sum coutline=black nolegend;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=global border;
```

Create the CUSTOMERS data set. The X and Y variables represent the unprojected longitude and latitude values in radians.

```
data customers;
  length city $20;
  input lastname$ zip x y city $;

  datalines;
Smith    27611 1.374164 0.623436 Raleigh
Jones    27560 1.375948 0.625278 Morrisville
Doe      27513 1.375279 0.624922 Cary
Patel    27520 1.369120 0.621970 Clayton
White    27705 1.377910 0.628629 Durham
Short    27587 1.370373 0.627680 WakeForest
Phillips 27591 1.368124 0.624705 Wendell
Jackson  27597 1.367264 0.625629 Zebulon
;
run;
```

Create the MYMAP data set of Wake County in North Carolina. Use the MAPSSAS libref to access the counties map data set. The FIPS county number for Wake is 183.

```
data mymap;
  set mapssas.counties(where=(fipstate(state)='NC' and county=183));
run;
```

Combine the CUSTOMERS and MYMAP data sets just created.

```
data combined;
  set customers mymap;
run;
```

Project the map and points data sets to both use Cartesian coordinates.

```
proc gproject data=combined out=combined dupok;
  id state county;
run;
```

Split the data.

```
data mymap customers;
  set combined;
  if missing(zip) = 1 then output mymap;
  else output customers;
run;
```

Determine which customer points fall inside or outside of the county. Note that GINSIDE can be run before or after the GPROJECT procedure.

```
proc ginside map=mymap data=customers out=customers;
  id state county;
run;
```

Print the resulting data.

```
proc print data=customers;
run;
```

Add annotations to the CUSTOMERS data set, using a green square for any customer data point inside the county map, and a red X if outside.

```
data customers;
   set customers;
   length color style $8;
   retain xsys ysys '2' hsys '3' when 'a' position '5' size 5;
   style='marker';
   if missing(county) then do;
     color='red';
     text='X';
     end;
   else do;
     color='green';
     text='U';       /* Marker font for a square */
     end;
   output;
   style='markere';  /* outline the symbols */
   color='gray33';
   output;
 run;
```

Define the title and map color for the graph.

```
title 'X Represents Customers Outside Wake County';
pattern1 v=s c=graydd;
```

Use the GMAP procedure to display the choropleth map, and annotate it with the variables in the CUSTOMERS data set. The ANNO= option specifies the annotate data set.

```
proc gmap data=mymap map=mymap anno=customers;
  id state county;
  choro county / stat=sum coutline=black nolegend;
run;
quit;
```

Chapter 7

GMAP Procedure

Overview: GMAP Procedure

About the GMAP Procedure

The GMAP procedure produces two-dimensional (choropleth) or three-dimensional (block, prism, and surface) maps that show variations of a variable value with respect to an area. Map data sets and response data sets are used in the GMAP procedure. A wide assortment of map data sets is available with SAS/GRAPH software. Map data sets based on the digital, vector-based maps from GfK GeoMarketing GmBH are available. These are in addition to the updated traditional map data sets provided by SAS.

Use the GMAP procedure to perform the following tasks:

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

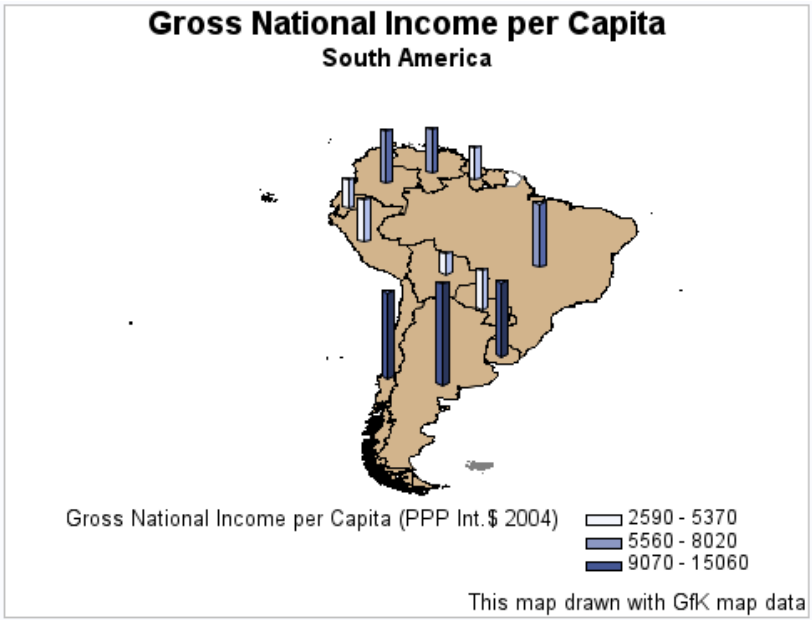
About Block Maps

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 is directly proportional to the value of the response variable.

Note: If the map area consists of multiple, noncontiguous areas, then the block is centered over the largest polygon of the set. For example, in the case of Japan the block is centered over Honshu, which is the largest island.

Figure 7.1 on page 190shows a simple block map of the gross national income per capita of countries in South America.. The gross national income per capita of each country (the response value) is represented by the height of the block.

Figure 7.1 *Block Map*



The program for this map is in "Example 1: Using GfK GeoMarketing Map Data to Produce a Simple Block Map" on page 281. For more information about producing block maps, see "BLOCK Statement" on page 222.

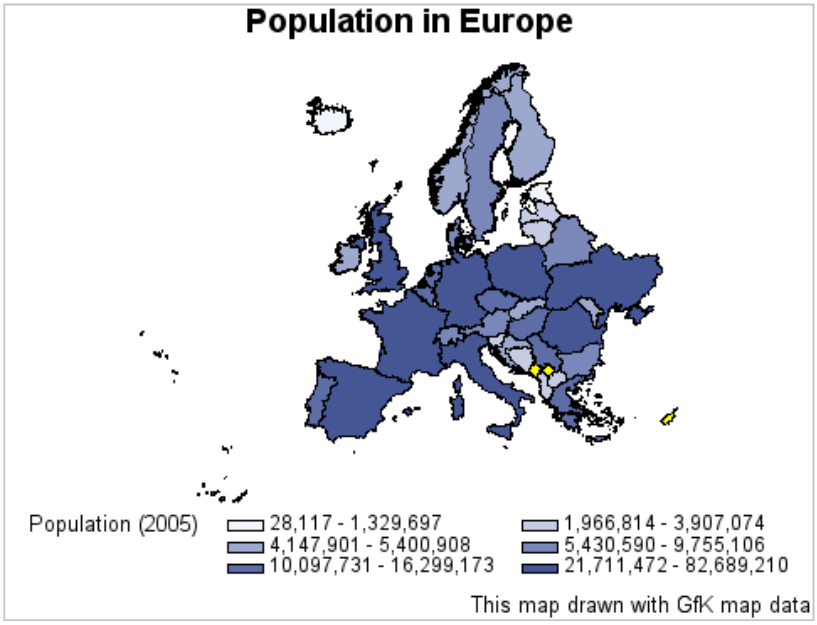
You can assign patterns to the areas in a block map by using the AREA statement. The values of the AREA variable are represented by the pattern of each map area. The values of the response variable in the BLOCK statement are represented by the height of the blocks. For more information, see "AREA Statement" on page 217.

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 7.2 on page 191 shows a choropleth map of the population of countries in Europe. The population of each country (the response value) is represented by the pattern that is assigned to the country.

Figure 7.2 *Two-dimensional (Choropleth) Map*



The program for this map is in "Example 9: Using GfK GeoMarketing Map Data to Produce a Simple Choropleth Map" on page 301.

You can also produce a simple choropleth map that shows an outline of a map's areas. Do this by specifying your map data set as both the map data set and the response data set in a GMAP statement and adding a PATTERN statement with VALUE=EMPTY. For more information about the PATTERN statement, see "PATTERN Statement" on page 44. For more information about producing choropleth maps, see "CHORO Statement" on page 235.

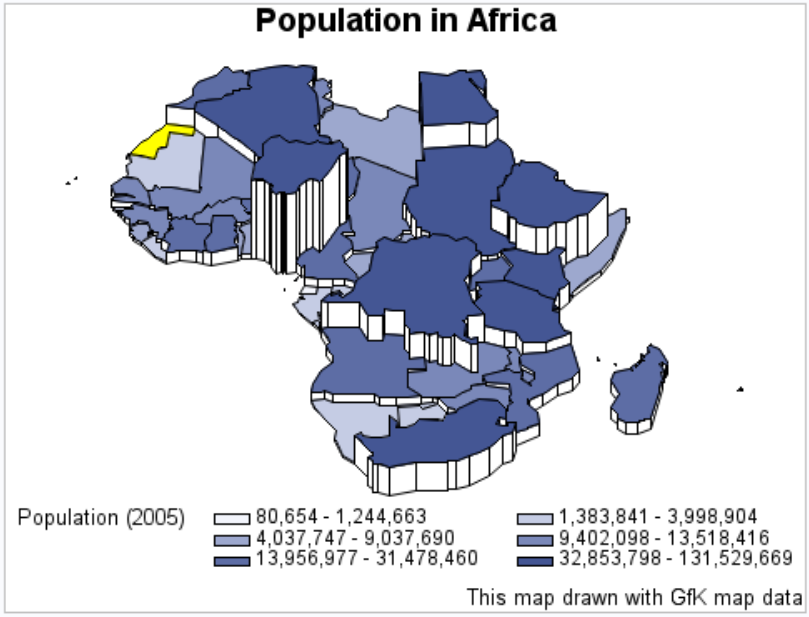
About Prism Maps

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, is directly proportional to the value of the response variable.

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 7.3 on page 192 shows a prism map of the populations of countries in Africa. The population of each country (the response value) is represented by the height of the country and the color of the country's map area.

Figure 7.3 *Prism Map*



The program for this map is in "Example 15: Using GfK GeoMarketing Map Data to Produce a Simple Prism Map" on page 316. For more information about producing prism maps, see "PRISM Statement" on page 245.

You can also assign patterns to the areas in a prism map by using the AREA statement. The values of the AREA variable are represented by the pattern of each map area. The values of the response variable in the PRISM statement are represented by the height of the map areas. For more information, see "AREA Statement" on page 217.

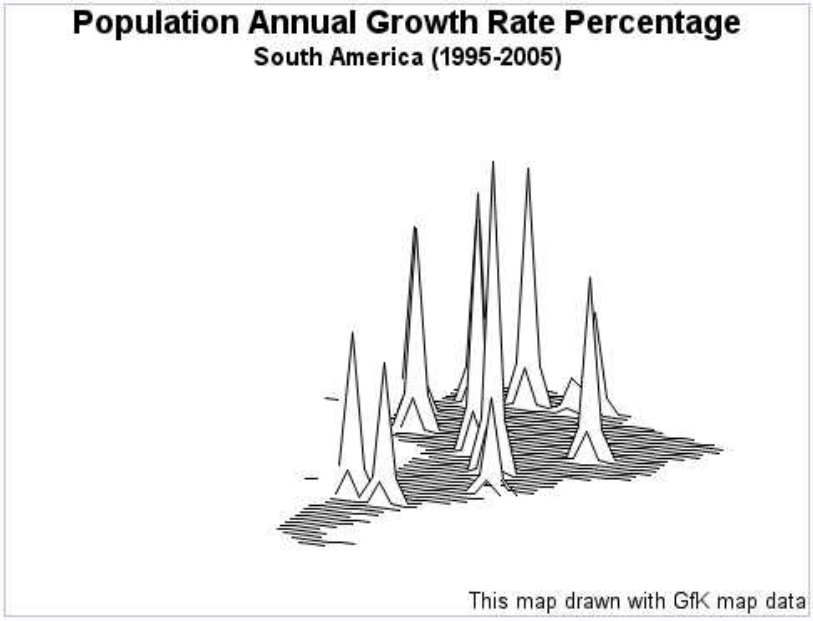
About Surface Maps

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 might 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 7.4 on page 193 shows a surface map of the population growth rates of countries in South America. The growth rate for each country (the response value) is represented by the height of the spike for that country.

Figure 7.4 *Surface Map*



The program for this map is in "Example 19: Using GfK GeoMarketing Map Data to Produce a Simple Surface Map" on page 325. For more information about producing surface maps, see "SURFACE Statement" on page 258.

About Map Data Sets

Map Data Set

A map data set is a data set that contains variables whose values are coordinates. These coordinates define the boundaries of map areas such as a state or country.

Map data sets are used in the GMAP procedure. Map data sets store spatial information across multiple observations. Each observation contains multiple variables with their specific data values. Map data sets supplied by SAS contain all the variables expected by the GMAP procedure. Map data sets that are not provided by SAS must contain X and Y boundary point coordinate variables as well as one or more map area identification variables. If these variables are missing, the procedure stops and displays an error message to the SAS log. The GMAP procedure can take as input a map data set and a response data set, provided that both data sets contain the same *id-variable*. Alternatively, you can use a single data set as input if it contains either the map data or a variable that references a map data set.

There are two types of data sets that are provided with SAS/GRAPH for mapping. GfK GeoMarketing digital, vector-based map data sets are available for use in addition to the traditional map data sets. All of the content in the traditional map data sets are represented in the GfK map data, and GfK also provides additional data. SAS licensed the map data from GfK GeoMarketing GmbH, and then converted the data into a SAS map data set format. The GfK map data sets are uniform and accurate for the whole world, and are intended to eventually replace the traditional map data sets.

SAS/GRAPH software includes a number of predefined map data sets.The traditional data sets are described in "The METAMAPS Data Set" on page 204.

About GfK GeoMarketing Map Data Sets

Background

GfK GeoMarketing digital, vector-based *map data sets* are available for use. These map data sets are located in the MAPSGFK library. The library reference (libref) MAPSGFK is set during system configuration and cannot be changed. Use the MAPS= system option within SAS to point to either the GfK map data set library MAPSGFK or the traditional map data set library MAPSSAS. MAPS points to the MAPSSAS library by default.

Updates to GfK Map Data Sets

Updates and changes to the GfK GeoMarketing map data sets are provided by GfK GeoMarketing GmbH. SAS makes these updates available via SAS Maps Online. See "Accessing SAS Maps Online" on page 210 for details about downloading data updates, sample SAS/GRAPH programs, and GIF images of maps.

About Traditional Map Data Sets

Background

Traditional *map data sets* have been available from the inception of SAS/GRAPH. These map data sets are resident in both the MAPS and MAPSSAS libraries. The library reference (libref) MAPSSAS is set during system configuration and cannot be changed. MAPSSAS is provided for ease-of-use when using the MAPS= system option within SAS. Use the option to point to either the traditional map data set library MAPSSAS or the GfK map data set library MAPSGFK.

Traditional Map Data Sets Containing Only X and Y

The traditional map data sets that contain X and Y variables (and no LONG and LAT variables), are usually projected maps. However, there are a few traditional map data sets for the US and Canada that contain X and Y values that are unprojected longitude and latitude. In this case, you need to use the GPROJECT procedure to project the map. (See Chapter 8, "GPROJECT Procedure," on page 355).

Note: You can determine whether a SAS traditional map data set is projected or unprojected in one of two ways. You can look at the description of each variable that is displayed when you use the CONTENTS procedure. You can also browse the MAPS.METAMAPS data set. An example of using the CONTENTS procedure follows:

```
proc contents data=maps.chile; /*<libref.datatsetname> */
run;
```

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

All except one of the GfK and most of the traditional map data sets that are provided with SAS/GRAPH software contain four coordinate variables (X, Y, LONG, and LAT). When all four coordinate variables are present, X and Y are always projected values that are used by the SAS/GRAPH procedures (by default). However, you might want to use the LONG and LAT variables to project the map again using a different projection type. If you need to use the unprojected values that are contained in the LONG and LAT variables, then do the following tasks:

1. Project the spherical coordinates (latitude and longitude) by using the GPROJECT procedure with its LATLON option, which uses the LAT and LONG variables instead of the Y and X variables.

   Note: LAT and LONG variables in GfK map data sets are in degrees. In traditional map data sets the LAT and LONG variables are in radians or degrees. Specify the DEGREES option to indicate that the LONG and LAT coordinates to project are degrees and not radians. Proj.4 projections use this by default.

   Note: In the GfK map data sets, the positive values of the LONG variable go to the east from the prime meridian. It is just the opposite in the traditional map data sets. Specifying the EASTLONG option when projecting traditional map data sets ensures that the positive values of the LONG variable go to the east from the prime meridian. Proj.4 projections use this by default.

   Note: Do not attempt to project MAPSGFK.US. This data set does not contain unprojected LAT and LONG variables.

2. Use the output data set from GPROJECT, which now contains Cartesian coordinates, as your input map data set by specifying it in the MAP= value in the GMAP procedure

Differences between GfK and Traditional Map Data Sets

This section covers the differences between the traditional and the GfK map data sets, and the benefits of using the GfK data. Usually, you cannot simply replace a traditional map data set in your existing code with the GfK map data set. You must carefully review the map libraries, the map data set filenames, and the variables that they contain. There are also notable projection differences. For example, the **X** and **Y** variables are always projected in the GfK map data sets. All the details are described next.

License Information

SAS supplies both traditional and GfK map data sets. SAS has licensed the vector-based map data sets representing the world from GfK GeoMarketing GmbH. These map data sets are to be used only with SAS/GRAPH for your internal business purposes.

Anyone with specialized map needs can license map data directly from GfK GeoMarketing GmbH. This map data matches up with the map data provided with

SAS/GRAPH. SAS customers can access GfK map data at http://www.gfk-geomarketing.com/en/digital_maps/softwarespecific_maps/maps_for_sas_graph.html.

The map data sets in library MAPSGFK are based on the digital, vector-based maps from GfK GeoMarketing GmbH and are covered by their copyright. For additional information, see http://support.sas.com/rnd/datavisualization/mapsonline/html/gfklicense.htm.

Libraries MAPSSAS and MAPS both contain the updated traditional maps data sets supplied by SAS.

Advantages to Using GfK Map Data Sets

There are some key advantages to using GfK instead of traditional map data sets, including:

Consistency
: Licensing the map data from GfK GeoMarketing GmbH provides a single source for map data. This single source ensures that the map data is accurate and uniform for the entire world. Additional map data obtained from GfK GeoMarketing GmbH matches seamlessly with SAS/GRAPH map data.

Single-source updates
: GfK GeoMarketing GmbH is solely responsible for all updates and changes to their map data. This includes political boundary updates, which were up to this time hard to obtain. SAS offers the ability to download the updates to map data via SAS Maps Online (http://support.sas.com/rnd/datavisualization/mapsonline/index.html).

Ease-of-Use
: SAS converts the GfK map data into a SAS map data set format to avoid unnecessary special processing.

Map Data Set Names

This section describes the differences found when comparing the GfK map data set names to the traditional map data set names.

Longer names
: GfK map data sets have longer names than the traditional map data sets. That is because they are not truncated. The names can exceed eight characters. For example, compare the traditional map data set name of **AFGHANIS** to the GFK map data set name of **AFGHANISTAN**.

Consistent naming convention
: The GfK map data sets also provide a consistent naming convention. For example, compare the traditional map data set name of **STATES**, which could correlate to any country with states, to the GFK map data set name of **US_STATES**.

  Another notable difference is the GfK map data set names of **NORTH_KOREA** and **SOUTH_KOREA** versus the traditional map data set names of **KOREANOR** and **KOREASOU**. GfK map data set names correspond to the common use names in language.

New data sets
: GfK includes new map data sets such as CAYMAN_ISLANDS and the Nomenclature of Territorial Units for Statistics (NUTS) level 0,1,2,3 data sets for Europe. The NUTS classification is a hierarchical system for dividing up the

economic territory of the European Union (EU). An example is the data set named EUROPENUTS3.

Data set designations
: Same-named data sets differentiated with numeric qualifiers, such as EUROPE, EUROPE1, and EUROPE2 indicate the level of administrative detail. For example, EUROPE1 indicates countries on the European continent with first administrative level—similar to US states.

  The level 1 continent map data sets do not contain all of the corresponding countries that are found in the MAPSGFK.WORLD data set. However, a level 0 continent map data set does contain all of its corresponding countries (for example, MAPSGFK.EUROPE).

New data set files with additional variables
: Each GfK map data set now has a companion data set with a **_ATTR** qualifier. For example, the data set AFRICA_ATTR contains extra variables.

New data set files listing all dependencies
: GfK data sets, where applicable, include an **_ALL** qualifier. These map data sets contain all the territories and islands. For example, compare the traditional map data set **US** that contains Puerto Rico with the GfK map data set of the same name, which does not contain Puerto Rico. You must use the GfK map data set **US_ALL** to include Puerto Rico.

Disputed territories
: In the **WORLD** map data set (MAPSGFK.WORLD), the disputed territories are not included with individual countries but rather are identified with the variable **ISOALPHA2** and a value of **NN**. They are also identified by the **ID** variable values that differ from the ISOAPLPHA2 variable value. For example, see the following ID values of the disputed territories between the following countries or states:

  - Suriname and Guyana has an ID value of **SR_GY**
  - China and Taiwan has ID value of **CN_TW**
  - China and India has ID value of **CN_IN**
  - Cameroon and Nigeria has ID value of **CM_NG**

  Note: Compared to MAPSGFK.WORLD map data set, the **ID** and **ISOALPHA2** variables in MAPSSGFK.WORLD_CITIES are the same.

New data set PROJPARM
: SAS provides this data set that contains the Procedure GPROJECT information for all GfK map data sets. This information includes values such as the Central Meridian or Standard Longitude, and the Parallel1 and Parallel2 values that were used during the projection procedure.

Correlations between Map Data Sets and Data Sets

The information from some traditional map data sets has been combined in the GfK map data sets. Some data set and map data set names remain the same between traditional and GfK. For example, the map data set **US** maintains the same name between traditional and GfK. The following table lists noteworthy correlations:

Table 7.1 *Notable Eliminated Map Data Sets and Data Sets*

Traditional (MAPSSAS)	GfK GeoMarketing (MAPSGFK)
USCITY *Note:* Includes towns, villages, hamlets, and other non-city areas that can have the same name as a city. A FEATYPE variable contains the area type.	USCITY *Note:* Includes towns, villages, hamlets, and other non-city areas that can have the same name as a city. Does not include a FEATYPE variable.
	USCITY_ALL *Note:* Includes Puerto Rico and the U.S. Virgin Islands.
COUNTIES	US_COUNTIES
COUNTY	US_COUNTIES
USCOUNTY	US_COUNTIES
CNTYNAME	US_COUNTIES_ATTR
STATES	US_STATES
US	US
US2	US_STATES_ATTR
USCENTER	USCENTER *Note:* Includes Washington, D.C.
	USCENTER_ALL *Note:* Includes Puerto Rico and the U.S. Virgin Islands.
All data set names beginning with USA	None

Note: The map data that was contained in traditional map data sets **COUNTIES**, **COUNTY**, and **USCOUNTY** is now combined in the one map data set **US_COUNTIES**.

Data set names with a numerical suffix (1, 2, 3) do not contain the same information between the traditional and GfK data sets. The MAPSGFK map data sets with a suffix of one (1), for example, contain countries with first administrative level—similar to U.S. states. The MAPSGFK map data sets with a suffix of two (2) contain countries with second administrative level - similar to U.S. counties.

Run the following code to determine whether the map data set name that you are currently using is listed in the MAPSGFK library:

```
proc datasets lib=mapsgfk;
run;
```

New and Changed Variables

The following table describes new variables and compares the changed variables between traditional and GfK map data sets:

Table 7.2 *Variables Specific to GfK Map Data Sets*

Variable Name	Variable Description	Traditional Map Data Set Details	GfK Map Data Set Details
DENSITY	Contains the density values returned from a GREDUCE procedure.	Might contain.	Does contain.
ID	A unique character code for a geographic area. For example, it represents a county or district in _ATTR map data sets, and a country in the MAPSGFK.WORLD _CITIES data set.	Might contain. (Numeric format)	Does contain. Character (Length 15 for all map data sets that contain this variable, regardless of the length of the value actually being stored)
IDNAME	A character code for a geographic area. For example, it represents a county or district in _ATTR map data sets, and a country in the MAPSGFK.WORLD _CITIES data set.	Does not contain.	Does contain.
IDNAMEU	A Unicode character version of **IDNAME** in _ATTR map data sets.	Does not contain.	Might contain.
ID1, ID2	A character state or province code in _ATTR map data sets.	Does not contain.	Might contain.
ID1NAME	A character state or province name in _ATTR map data sets.	Does not contain.	Might contain.
ID1NAMEU	A Unicode character version of **ID1NAME** in _ATTR map data sets.	Does not contain.	Might contain.

Variable Name	Variable Description	Traditional Map Data Set Details	GfK Map Data Set Details
ISO	A character country code in _ATTR map data sets.	Does not contain.	Does contain.
ISOALPHA2, ISOALPHA3	A character country International Organization for Standardization Alpha2– or Alpha3– code in _ATTR map data sets. *Note:* **ISOALPHA2** is used in the MAPSGFK.WORLD map data set with a value of **NN** to indicate a disputed territory.	Does not contain.	Does contain.
ISONAME	A character country International Organization for Standardization name in _ATTR map data sets. *Note:* the **ISONAME** variable value found in a continent _ATTR map data set is identical to its counterpart in the MAPSGFK.WORLD map data set	Does not contain.	Does contain.
LAT	A numeric variable containing the vertical coordinate of the boundary point. (The value of this variable is unprojected and represents latitude (north-south position).	Might contain. (The value of the variable is in radians or degrees.)	Does contain. (The value of the variable is in degrees.)
LONG	A numeric variable containing the horizontal coordinate of the boundary point. (The value of this variable is unprojected and represents longitude (east-west position).	Might contain. (The value of the variable is in radians or degrees. Positive values indicate west.)	Does contain. (The value of the variable is in degrees. Positive values indicate east.)

Variable Name	Variable Description	Traditional Map Data Set Details	GfK Map Data Set Details
RESOLUTION	A numeric map detail level from 1 to 10 that is based on desired output display resolution.	Does not contain.	Does contain.
STATE *Note:* Variable occurs only in U.S. data sets	State FIPS code.	Might contain. (Numeric FIPS code.)	Might contain. (Character identification.)
X	A numeric variable that contains the horizontal coordinates of the boundary points.	Does contain. (The longitudinal value can either be projected or unprojected.	Does contain. (The value is always projected and represents longitude.)
Y	A numeric variable that contains the vertical coordinates of the boundary points	Does contain. (The latitudinal value can either be projected or unprojected.)	Does contain. (The value is always projected and represents latitude.)

Note: Apply the $uesc50. format to any Unicode variable to view the national characters in an English SAS UTF8 session. The following example formats the IDNAMEU variable:

```
data my_attr;
set mapsgfk.reunion_attr;
Unicode_idname=input(IDNAMEU, $uesc50.);
run;

goptions ftext='Albany AMT';
proc gmap data=my_attr map=mapsgfk.reunion;
id id;
choro Unicode_idname;
run;
quit;
```

Eliminated Variables

The following table describes the traditional map data set variables that are not used in the GfK map data sets, as well as their replacement, where applicable, in the GfK map data sets:

Variable Name	Variable Description	Traditional Map Data Set Details	GfK Map Data Set Details
CDCODE	Census District.	Does contain. (Character format)	Does not contain.

Variable Name	Variable Description	Traditional Map Data Set Details	GfK Map Data Set Details
CONT	Continent.	Does contain. (Numeric format) *Note:* Used in all the continent and WORLD map data sets.	Does contain. (Numeric format) *Note:* Used in all the *continent*_ATTR, WORLD, and WORLD_ATTR map data sets.
COUNTRY	World Geographic Code.	Might contain. (Numeric format)	Does not contain. Replaced by the **ISO** variable that represents the International Organization of Standardization's country code. (Character format)
ID	Identification variable.	Might contain. (Numeric format).	Does contain. (Character format)
PROVINCE	Character abbreviation for a province.	Might contain. (Character format)	Does not contain. Replaced by the **ID** variable in character format.
GLC codes	Geographic Locator Codes (alphanumeric).	Might contain.	Does not contain. Replaced by the ISO codes.

Data Changes

The data changes encompassed in the GfK map data sets include various variables. The changes are detailed in this section.

The **RESOLUTION** variable reduces the points for the display size of a map as indicated in the following table:

Table 7.3 *RESOLUTION Variable Values*

RESOLUTION VALUE	NUMBER OF PIXELS
1	320 x 240
2	400 x 300
3	640 x 480
4	800 x 600

RESOLUTION VALUE	NUMBER OF PIXELS
5	1280 x 1024
6	1600 x 1200
7	2400 x 1800
8	6000 x 4800
9	14400 x 11520
10	28800 x 23040

In data sets with one (1) administrative level, the ID variable concatenates the country code and the administrative_level code. In data sets with two (2) administrative levels, the ID variable concatenates the country code and the lower_administrative_level code. The ID1 variable concatenates the country code and the higher_administrative_level code. Some data sets have three (3) administrative levels. In this case the ID variable concatenates the country code and lowest_administrative_level code. The ID2 variable concatenates the country code and middle_administrative_level code. The ID1 variable concatenates the country code and highest_administrative_level code.

Here are some specific examples. All the traditional U.S. map data sets contain a **STATE** variable. The ID variable in the MAPSGFK US_STATES data set represents a value for state, such as US-37 for NC. However, ID in the US_COUNTIES data set represents a county value, such as US-37183 for Wake County, NC. In the MAPSGFK US_COUNTIES data set the ID1 variable contains the value for state, for example, US-37 for North Carolina. **ID1** concatenates the country code with the state code.

The traditional **COUNTY** variable is represented as **ID** in GfK map data sets such as **US_COUNTIES**. For example, a **COUNTY** value of **15** in California (CA) is represented by **ID** as **US-06015**. **ID** concatenates the country code, the state code, and the county code.

The X and Y variable values in the GfK map data sets are always projected.

The LAT and LONG variable values in the GfK map data sets are always unprojected degrees (not radians).

The identification variables (for example, ID and ID1) are character instead of numeric in the GfK map data sets.

GfK map data is provided in UTF-8 Unicode encoding with the exception of the MAPSGFK.WORLD_CITIES map data set.

GfK map data contained in the map data sets are projected using appropriate projection methods. For example, the data set MAPSGFK.US represents a relatively small map area that is near neither pole. These factors make it possible to project data without distortion using the Albers method. The **LABEL** column in each data set should identify the projection algorithm used for the X and Y variables. Use the CONTENTS procedure to view columns in a map data set. To view columns in the map data set named MAPSGFK.US, run the following code:

```
proc contents data=mapsgfk.us;
run;
```

Compatibility Information

Substituting the GfK map data sets into an existing application that currently uses the traditional map data sets requires modifications to the application. Without these modifications, using the GfK map data could cause unexpected results. For example, the GfK map data set names should not be substituted into your existing code's DATA= statement without some additional considerations. Please refer to "Migration Information" on page 204 for tips on converting your existing map-producing code.

Migration Information

When using the GfK map data sets, be aware of the following:

- the map data set filenames can be different
- the **X** and **Y** variables are always projected.
- the unprojected **LAT** and **LONG** variables are in degrees and not radians.
- the positive values of the LONG variable go east from the prime meridian. In the traditional map data sets, the positive values go west.
- the **ID**, **ID1**, and **ID2** variables are character instead of numeric format.
- A map area crossing the International Date Line is divided into two segments in the displayed graph, indicating the boundaries on either side of the line. Examples of affected regions are Russia, Tonga, and territories and islands in the Pacific ocean.

Refer to "Reworking Code That Uses Traditional Map Data Sets" on page 278 for information and tips.

The METAMAPS Data Set

The maps data set library that is referenced by both the MAPS and MAPSSAS librefs have a data set named METAMAPS, which contains metadata about all of the data sets that are delivered in the library. MEMNAME is just one of the variables to find among the metadata in MAPS.METAMAPS and MAPSSAS.METAMAPS:

Variable
: Description

VAR**x**
: identifies the name of the variable in the data set, where **x** is a numerical value.

LABEL**x**
: describes **VARx** in the data set.

MEMNAME
: identifies the names of all of the data sets that are delivered in the MAPS or MAPSSAS library.

Special GfK Data Sets for Annotating Maps

There are several data sets in the MAPSGFK library that enable you to easily label maps. Some of their names differ from any traditional data sets that you might have used. Though the non-map data sets contain coordinates for map features such as cities, they cannot be used as map data sets.

The X and Y variables containing projected coordinates in the USCENTER, USCENTER_ALL, USCITY, and USCITY_ALL data sets can be used directly with the map data sets specified in the file header. If you want to use a different map data set, then use the LAT and LONG variables provided in the MAPSGFK data set. Use the Chapter 8, "GPROJECT Procedure," on page 355 to project them using the map data set projection parameters.

MAPSGFK.USCENTER_ALL
: contains the coordinates of the visual center of each state in the U.S. and Washington, D.C., Puerto Rico, and the U.S. Virgin Islands. It also contains coordinates in the ocean for states that are too small to contain a label. There are two pairs of variables for locating labels using Annotate data sets. The X and Y variables contain projected coordinates and can be used with the MAPSGFK.US_ALL data set. The LONG and LAT variables contain the unprojected longitude and latitude in degrees (not radians). These can be used to place labels on any US data set, noting that you can subset the data set by state as needed.

MAPSGFK.USCENTER
: is similar to USCENTER_ALL, except that it does not include Puerto Rico and the U.S. Virgin Islands. The X and Y variables contain projected coordinates and can be used with the MAPSGFK.US data set.

MAPSGFK.USCITY_ALL
: contains the locations of selected cities in the U.S. Many city names occur in more than one state, so you might have to subset by state to avoid duplication. There are two pairs of variables for locating labels using Annotate data sets. The X and Y variables contain projected coordinates using the Miller II projection method. These variables can be used with the MAPSGFK.US_ALL data set. The LONG and LAT variables contain the unprojected longitude and latitude in degrees (not radians). These can be used to place labels on any US data set, noting that you can subset the data set by state as needed.

MAPSGFK.USCITY
: is similar to USCITY_ALL, except that it does not include Puerto Rico and the U.S. Virgin Islands. The X and Y variables contain projected coordinates using the Albers projection method. These variables can be used with the MAPSGFK.US data set.

  Duplicate observations can be listed in the output data set. This occurs because the map data set includes towns, villages, hamlets, and other non-city areas that can share the same name with a city.

MAPSGFK.US_COUNTIES
: contains a projected US county map with Alaska and Hawaii in physical locations. Puerto Rico is not included. Many county names occur in more than one state, so you might have to subset by state to avoid duplication. The LONG and LAT variables contain the unprojected longitude and latitude in degrees (not radians). These can be used to place labels on any US data set, noting that you can subset the data set by state as needed.

MAPSGFK.US_STATES
contains a projected US state map with Alaska and Hawaii in physical locations. Puerto Rico is not included. The LONG and LAT variables contain the unprojected longitude and latitude in degrees (not radians). These can be used to place labels on any US data set, noting that you can subset the data set by state as needed.

MAPSGFK.US
contains a projected US state map with Alaska and Hawaii inset into the map. Puerto Rico is not included. The LONG and LAT variables are not included.

MAPSGFK.US_ALL
contains a projected US county map with Alaska and Hawaii in physical locations. Puerto Rico and the United States Virgin Islands are included. Many county names occur in more than one state, so you might have to subset by state to avoid duplication. The LONG and LAT variables contain the unprojected longitude and latitude in degrees (not radians). These can be used to place labels on any US data set, noting that you can subset the data set by state as needed.

MAPSGFK.WORLD_CITIES
contains the location of more than 200,000 international cities. Many city names occur more than once, so you might want to specify a country, state, province, or region variable to differentiate the cities. This data set does contain local city spellings. The CITY variable has mixed case values. The CITY2 variable values in this data set are normalized. This means that they are uppercase and stripped of all spaces and characters that are not alphabetic or numeric. This CITY2 variable is used by the GEOCODE procedure. The X and Y variables contain projected coordinates. The LONG and LAT variables contain the unprojected longitude and latitude in degrees (not radians). These can be used to place annotation labels on any international data set.

Note: MAPSGFK.WORLD_CITIES_ALL is an alternate data set that contains the location of more than a million cities worldwide. It contains the same variables as the MAPSGFK.WORLD_CITIES data set. The more comprehensive data set is available for download from the SAS Maps Online website. This unabridged data set is updated annually.

Special Traditional Data Sets for Annotating Maps

There are several data sets in the MAPSSAS library that enable you to easily label maps. These data sets contain coordinates for map features such as cities, but cannot be used as map data sets.

MAPS.USCENTER
contains the coordinates of the visual center of each state in the U.S. and Washington, D.C.. It also contains coordinates in the ocean for states that are too small to contain a label. There are two pairs of variables for locating labels using Annotate data sets. The X and Y variables are projected and can be used with the MAPS.US and MAPS.USCOUNTY data sets. The LONG and LAT variables are unprojected longitude and latitude in radians and can be used with the MAPS.STATES, MAPS.COUNTIES, and MAPS.COUNTY data sets.

MAPS.USCITY
contains the locations of selected cities in the U.S. Many city names occur in more than one state, so you might have to subset by state to avoid duplication. There are two pairs of variables for locating labels using Annotate data sets. The X and Y variables contain projected coordinates and can be used with the MAPS.US and

MAPS.USCOUNTY data sets. The LONG and LAT variables contain the unprojected longitude and latitude in radians. These can be used to place labels on the MAPS.STATES, MAPS.COUNTIES, or MAPS.COUNTY data sets.

Duplicate observations can be listed in the output data set. This occurs because the map data set includes towns, villages, hamlets, and other non-city areas that can share the same name with a city. Differentiate the output with the FEATYPE variable, which contains the area type.

For details about each of these data sets, see the MAPS.METAMAPS data set or the MAPSSAS.METAMAPS data set.

About Response Data Sets

Definition: Response Data Set

A *response data set* is a SAS data set that can contain other variables, but requires the following variables:

- 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.

Using the Response Data Set with the Map Data Sets

The map data set and the response data set must be used independently in the PROC GMAP statement. The response data set is specified by the DATA= option and the map data set is specified by the MAP= option. The values of the map area *id-variables* in the response data set determine the map areas to be included on the map. Unless the ALL option is used 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, then create a subset of the map data set for efficiency.

If you have a response data set named SASHELP.US_DATA, then the syntax for using the GMAP procedure might resemble the following:

```
proc gmap map=mapsgfk.us data=sashelp.us_data;
   id state;
   choro region;
run;
quit;
```

SAS provides sample response data sets in the SASHELP library. These data sets are available for you to use for examples and for testing code. Some of the examples in this chapter use sample response data sets such as SASHELP.US_DATA and SASHELP.DEMOGRAPHICS.

About Response Variables

The GMAP procedure can produce block, choropleth, prism, and surface 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 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 be determined by any of the following:

- a character value
- the MIDPOINTS= option
- a range of numeric values
- a specific numeric value

When response levels are determined by a character value, the GMAP procedure treats each unique value as a response level. For example, if the response variable contains the names of ten regions, each region is a response level, resulting in ten response levels.

When character response levels are determined by the MIDPOINTS= option, any response variable values that do not match one of the specified response level values are ignored.

When response levels are determined by a range of numeric values, each response level has a similar number of observations. These options are exceptions to this:

- The LEVELS= option specifies the number of response levels to be graphed for the response variable. The LEVELS=*number-of-midpoints* option is ignored if either the DISCRETE or MIDPOINTS= option is used.
- 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, then 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.

When response levels are determined by specific numeric values, and the DISCRETE option is specified, one level is created for each value. If the response variable has an associated format, then each formatted value is represented by a different response level.

The AREA, 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. If you specify the AREA statement on a BLOCK map, then the response levels for AREA variable are shown as map areas. 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 44 for more information about fill pattern values and default pattern rotation.

About Identification Variables

For map data sets and response data sets, *id-variables* 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. It does this in order to create the display 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. The following table describes the conditions under which the procedure does or does not display map areas and response data.

Table 7.4 *Displaying Map Areas and Response Data*

Characteristic of the response data set	Additional variables	Result
Includes the map area.	The map area has a response value.	The procedure draws the map area and displays the response data.
Includes the map area.	The response value for the map area is a missing value.	The procedure draws the map area but leaves it empty.
Includes the map area.	The response value for the map area is a missing value and the MISSING option is used in the map statement.	The procedure draws the map area and displays a response level for the missing value.

Characteristic of the response data set	Additional variables	Result
Includes the map area.	The map area has a response value and the MAP= option and the DATA= statement specify the same map data set.	The procedure draws the map area and displays boundary outlines but does not display the value of the response variable or variables.
Does not include the map area.	The ALL option is used in the PROC GMAP statement.	The procedure draws the map area but leaves it empty.
Does not include the map area.	The ALL option is not used.	The procedure does not draw the map area.

Summary of Use

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

1. When using a map data set, 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.
2. 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.
3. When using a map data set, use the PROC GMAP statement to identify the map data set as the MAP= value and response data set as the DATA= value.
4. Use the ID statement to specify the *id-variable(s)*.
5. Use a BLOCK, CHORO, PRISM, or SURFACE statement to identify the response variable and generate the map.

Accessing SAS Maps Online

Visit SAS Maps Online to download data updates, sample SAS/GRAPH programs that use the production-level map data sets delivered with SAS/GRAPH, and GIF images of maps. SAS Maps Online is located at the following URL: http://support.sas.com/rnd/datavisualization/mapsonline/index.html

After downloading and unzipping map data sets, you must take them out of transport format by running the CIMPORT procedure using your current version of SAS. For more information, see "Transporting and Converting Graphics Output" in *SAS/GRAPH: Reference*.

Importing Maps from Esri Shapefiles

You can import Esri shapefiles as map data sets by using the MAPIMPORT procedure. Depending on the type of coordinates that are in your shapefile, you might want to perform additional processing. For example, you might want to project the map with the GPROJECT procedure, or use the GREDUCE procedure to create a DENSITY variable for reducing your data.

For more information, see Chapter 11, "MAPIMPORT Procedure," on page 417.

Syntax: GMAP Procedure

Requirement:
- An input map data set is required. If not specified, an input map data set must have been created before issuing the PROC GMAP statement.
- A response data set is required. If not specified, a response data set must have been created before issuing the PROC GMAP statement.
- One ID statement
- At least one CHORO, BLOCK, PRISM, or SURFACE statement

Global statements: TITLE, FOOTNOTE, NOTE, LEGEND, PATTERN, GOPTIONS

Supports: RUN-group processing

Note: The procedure can include the SAS/GRAPH statements BY on page 19 and NOTE on page 56, as well as the Base SAS statements FORMAT, LABEL, and WHERE. See "SAS/GRAPH Statements" in *SAS/GRAPH: Reference* and *SAS Statements: Reference* for more information.

PROC GMAP MAP=*map-data-set*
DATA=*response-data-set*
<ALL>
<ANNOTATE=*Annotate-data-set*>
<DENSITY=0 ...6 | LOW | MEDIUM | HIGH>
<GOUT=<*libref.*>*output-catalog*>
<IMAGEMAP=*output-data-set*>
<LATLON>
<RESOLUTION=1 ...10 | AUTO | NONE>
<STRETCH>
<UNIFORM>;

ID *id-variable(s)*;

AREA *response-variable* <| *option(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 contain the variables associated with the map. The statement also provides the option to display all map areas and to specify annotation and an output catalog.

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

Syntax

PROC GMAP MAP=*map-data-set*
DATA=*response-data-set*
<ALL>
<ANNOTATE=*Annotate-data-set*>
<DENSITY=0 ...6 | LOW | MEDIUM | HIGH>
<GOUT=<*libref.*>*output-catalog*>
<IMAGEMAP=*output-data-set*>
<LATLON>
<RESOLUTION=1 ...10 | AUTO | NONE>
<STRETCH>
<UNIFORM>;

Summary of Optional Arguments

ALL
: specifies that the maps generated by the procedure should include all of the map areas from the map data set. This occurs even if the response data set does not include an observation for the map area.

ANNOTATE=*Annotate-data-set*
: specifies a data set to annotate all of the maps that are produced by the GMAP procedure.

DENSITY=0 …6 | LOW | MEDIUM | HIGH
: for maps that have a DENSITY variable, specifies the density of map observations that are used.

GOUT=<*libref.*>*output-catalog*
: specifies the SAS catalog in which to save the graphics output that is produced by the GMAP procedure for later replay.

IMAGEMAP=*output-data-set*
: creates a temporary SAS data set that is used to generate an image map.

LATLON
: specifies that the unprojected LAT and LONG variables from the map data set are used for coordinate data instead of the Y and X variables.

RESOLUTION=1 …10 | AUTO | NONE
: for maps that have a RESOLUTION variable, specifies the resolution level of map observations that are used.

STRETCH
: stretches map extents to cover all available space in the device. This might cause the map to be distorted.

UNIFORM
: causes the same legend and coloring to be used for all maps produced by the procedure instead of being calculated within each BY group for each map.

Required Arguments

DATA=*response-data-set*

identifies the SAS data set that contains the response values or the response values and the spatial information that are evaluated and represented on the map. If not specified, a response data set must have been created before issuing the PROC GMAP statement.

Default The GMAP procedure uses the most recently created SAS data set.

Requirements This statement is required if no response data set was created before issuing the PROC GMAP statement.

A response data set must contain the same identification variable or variables as the specified map data set, along with the values of the response variable.

See "The SAS Data Set: Your Key to the SAS System" in *Step-by-Step Programming with Base SAS*

"About Data Set Options" in *SAS Data Set Options: Reference*

MAP=*map-data-set*

names a map data set that contains either the X and Y or LAT and LONG coordinates, or all, for the boundary points of each map area. This includes map data sets licensed by SAS from GfK GeoMarketing GmBH and converted by SAS to a SAS data set format.

Requirement This statement is required.

Interaction The map data set must contain the same identification variable or variables as the response data set being used.

See "About Map Data Sets" on page 193

Optional Arguments

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

ALL

specifies that the maps generated by the procedure should include all of the map areas from the map data set. This occurs even if the response data set does not include an observation for the map area. When you use the ALL option and a BY statement in a RUN group, the maps generated for each BY group include every map area from the map data set.

See "Displaying Map Areas and Response Data" on page 209

ANNOTATE=*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 the ANNOTATE= option in the action statement.

Alias ANNO=

See "Using Annotate Data Sets" in *SAS/GRAPH: Reference*

DENSITY=0 …6 | LOW | MEDIUM | HIGH

for maps that have a DENSITY variable, specifies the density of map observations that are used. The value that you specify indicates the maximum value that the DENSITY variable can have for the observation to be displayed. For example, if you specify DENSITY=5, then only observations in the map data set whose DENSITY value is less than or equal to 5 are displayed.

Intuitively, the DENSITY variable specifies how close a map point is to other map points. If there are many map points in close proximity (high density), then it is possible to eliminate a number of them without seriously degrading the quality of the map. Many map data sets supplied by SAS contain a DENSITY variable. For map data sets that do not contain a DENSITY variable, you can add and populate the variable using the GREDUCE procedure.

You can specify an integer from 0 to 6 for the DENSITY option, or you can specify one of the descriptors. LOW is equivalent to DENSITY= 1. MEDIUM is equivalent to DENSITY= 3, HIGH is equivalent to DENSITY= 6.

If you do not specify the DENSITY= option, nor a RESOLUTION= option, then this is the same as specifying RESOLUTION= AUTO. All of the observations in a map data set are displayed that correspond to the resolution that the device displaying the map is using. This is regardless of whether the input map data set contains a DENSITY variable or not.

Default DENSITY=6

Restriction If the map data set does not contain a column of DENSITY values, then a warning is issued and the option is ignored.

Interactions When the DENSITY= option is specified, and the RESOLUTION= option is not, the DENSITY= option is used. If the DENSITY= option is not specified, then the RESOLUTION= option is used automatically, regardless of whether it is actually specified. If specified, the RESOLUTION= setting is used. Otherwise, the RESOLUTION= value is set to the numeric level corresponding to the resolution that the device displaying the map is using.

If you specify both the RESOLUTION= and DENSITY= options, and those variables exist in the input map data set, then the RESOLUTION= option takes precedence.

Tip Specifying DENSITY=0 is not equivalent to not specifying a DENSITY= option.

See Chapter 9, “GREDUCE Procedure,” on page 389 for information about the DENSITY variable.

“RESOLUTION=1 …10 | AUTO | NONE” on page 215

Example “Example 15: Using GfK GeoMarketing Map Data to Produce a Simple Prism Map” on page 316

GOUT=*<libref.>output-catalog*

specifies the SAS catalog in which to save the graphics output that is produced by the GMAP procedure for later replay. You can use the GREPLAY procedure to view the graphs stored in the catalog. If you do not use the GOUT= option, catalog entries are written to the default catalog WORK.GSEG, which is erased at the end of your session.

Restriction Not supported by Java and ActiveX

See "Specifying the Catalog Name and Entry Name for Your GRSEGs " in *SAS/GRAPH: Reference*

IMAGEMAP=*output-data-set*

creates a temporary SAS data set that is used to generate an image map in an SVG file when you are sending output to the LISTING destination. (This option is not necessary when you are sending output to the HTML destination.) The drill-down URLs in the image map must be provided by variables in the input data set. These variables are identified to the procedure with the HTML= and HTML_LEGEND= options.

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" in *SAS/GRAPH: Reference* and "Enhancing Drill-Down Behavior in SVG Presentations Using HTML Attributes" in *SAS/GRAPH: Reference*

LATLON

specifies that the unprojected LAT and LONG variables from the map data set are used for coordinate data instead of the Y and X variables. The LAT and LONG variables represent the unprojected latitude and longitude coordinates, respectively. All GfK GeoMarketing and most traditional map data sets contain both sets of variables (Y, X, and LAT, LONG) for projected and unprojected maps. By default, the Y and X variables are used to produce a projected map. However, when LATLON is specified, the Y and X variables are no longer required by the GMAP procedure.

Alias LATLONG

Example "Example 4: Projecting an Annotate Data Set" on page 384

RESOLUTION=1 …10 | AUTO | NONE

for maps that have a RESOLUTION variable, specifies the resolution level of map observations that are used. The numeric value that you specify indicates the maximum value that the RESOLUTION variable can have for the observation to be displayed. For example, if you specify RESOLUTION=5, then only observations in the map data set whose RESOLUTION value is less than or equal to 5 are displayed.

The RESOLUTION variable specifies the screen resolution at which to display a map point. If there are many map points in close proximity (high density), then it is possible to eliminate a number of them without seriously degrading the quality of the map. Many map data sets supplied by SAS contain a RESOLUTION variable.

You can specify an integer from 1 to 10 for the RESOLUTION option, or you can specify AUTO or NONE. The integer correlates to the number of pixels displayed on the map, A low resolution value indicates that a smaller number of pixels is displayed. For example, a RESOLUTION value of 4 equates to 800 x 600 pixels displayed. Specifying AUTO defaults to the resolution setting of the device being used in the GMAP procedure. A RESOLUTION set to NONE indicates that the DENSITY option, if specified, is used instead.

If you do not specify the RESOLUTION= option, nor a DENSITY= option, then this is the same as specifying RESOLUTION= AUTO. All of the observations in a map data set are displayed that correspond to the resolution that the device displaying the map is using. This is regardless of whether the input map data set contains a RESOLUTION variable or not.

Alias RES=

Default RESOLUTION=AUTO when the DENSITY option is not specified. When the DENSITY option is specified, RESOLUTION=NONE.

Restriction If the map data set does not contain a column of RESOLUTION values, then a warning is issued and the option is ignored.

Interactions When the DENSITY= option is specified, and the RESOLUTION= option is not, the DENSITY= option is used. If the DENSITY= option is not specified, then the RESOLUTION= option is used automatically, regardless of whether it is actually specified. The RESOLUTION= value is set to the level corresponding to the resolution that the device displaying the map is using.

If you specify both the RESOLUTION= and DENSITY= options, and those variables exist in the input map data set, then the RESOLUTION= option is used. In this case the DENSITY= option is ignored.

See Table 7.3 on page 202 for the correlation between the RESOLUTION value and the number of pixels displayed.

Example "Example 1: Using GfK GeoMarketing Map Data to Produce a Simple Block Map" on page 281

STRETCH

stretches map extents to cover all available space in the device. This might cause the map to be distorted. When this option is applied to the PROC GMAP statement, it applies to all statements. If applied to a single statement, it applies only to that statement.

Restriction Not supported by Java and ActiveX.

UNIFORM

causes the same legend and coloring to be used for all maps produced by the procedure instead of being calculated within each BY group for each map. The UNIFORM option pre-scans the data to generate a categorization across all the data, regardless of BY grouping, and applies that categorization to all maps in the BY group. This results in all maps uniformly displaying legends and coloring such that a single value always has the same color in multiple maps.

When specified in a PROC GMAP statement, UNIFORM applies to all BLOCK, CHORO, AREA, and PRISM statements included within the GMAP run-group.

When omitted from the PROC GMAP statement, and specified on an individual BLOCK, AREA, CHORO, or PRISM statement, UNIFORM applies only to the maps produced by that statement.

Restriction Not supported by Java.

ID Statement

Identifies the variable or variables in the input data set(s) that define map areas.

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

Syntax

ID *id-variable(s)*;

Required Argument

id-variable(s)

identifies one or more variables in the map and response data sets that define a map area. This argument is used only when map and response data sets are specified.

Every variable that is listed in the ID statement must appear in both the map and response data sets. The variable identified by the *id-variable(s)* argument can be of type numeric or character and should have the same name, type, and length in both the response and map data sets.

Note: If the *id-variables* in the response data set and map data set do not have the same length, then your map areas might not be drawn correctly.

See "About Identification Variables" on page 209

Examples "Example 1: Using GfK GeoMarketing Map Data to Produce a Simple Block Map" on page 281

"Example 5: Using GfK GeoMarketing Map Data to Assign a Format to the Response Variable" on page 290

"Example 9: Using GfK GeoMarketing Map Data to Produce a Simple Choropleth Map" on page 301

AREA Statement

applies color to the regions in BLOCK and PRISM maps based on values of a specified response variable.

Requirement: The response variable is required. The AREA statement must be used in conjunction with either a BLOCK or PRISM statement.

Syntax

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

Summary of Optional Arguments

DISCRETE
: generates a separate response level (color and surface pattern) for each different value of the formatted response variable.

LEGEND=LEGEND<1 …99>
: specifies the LEGEND statement to associate with the map.

LEVELS=*number-of-response-levels* | ALL
: specifies the number of response levels to be graphed for the response variable.

MIDPOINTS=*value-list* | OLD

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

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

NOLEGEND
suppresses the legend.

PERCENT
causes GMAP to collect all response values (or their statistic) and chart each region as a percentage of the whole.

RANGE
causes GMAP to display, in the legend, the starting value and ending value of the range around each midpoint specified with the MIDPOINTS= option (instead of displaying just the midpoints).

STATFMT=*format-specification*
overrides the GMAP default format for percent of PERCENT8.2.

STATISTIC=FIRST | SUM | FREQUENCY | MEAN
specifies the statistic for GMAP to chart.

UNIFORM
causes the same legend and coloring to be used for all maps produced by the procedure instead of being calculated within each BY group for each map.

Required Argument

response-variable
specifies the variable in the response data set. Areas that correspond to response variables with missing values are not colored unless you use the MISSING option in the AREA statement. This variable is represented in all BLOCK and PRISM maps in the same RUN group.

See "About Response Variables" on page 208

Optional Arguments

Options in an AREA 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. All of these options are the same as the normal GMAP options except that they apply to the areas of regions only, and not to the bar heights.

DISCRETE
generates a separate response level (color and surface pattern) for each different value of the formatted response variable.

The LEVELS=*number of midpoints* option is ignored when you use the DISCRETE option.

If you specify the DISCRETE option, then distinct, non-continuous colors are used for the response values. If you specify the LEVELS= option instead, then a color ramp is used to assign each response value a continuous color scheme.

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

LEGEND=LEGEND<1 …99>
specifies the LEGEND statement to associate with the map. The LEGEND= option 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. Most of the LEGEND options described in "LEGEND Statement" on page 25 are supported by both Java and ActiveX. If a LEGEND option is not supported by Java or ActiveX, it is noted in the LEGEND option definition.

Restriction Partially supported by Java and ActiveX

See "LEGEND Statement" on page 25.

LEVELS=*number-of-response-levels* | ALL
specifies the number of response levels to be graphed for the response variable.

Use LEVELS= to specify the number of response variables when the response variables are numeric. When you specify LEVELS=ALL, all unique numeric or character response variable values are graphed.

Each response level is assigned a different surface pattern and color combination. The prism and block heights are based on the data value of the corresponding response variable.

If you specify the LEVELS= option, then a color ramp is used to assign each response value a continuous color scheme. The response values are assigned lighter and darker values of a color scheme to express lower and higher response values. If you specify the DISCRETE option, then distinct, non-continuous colors are used for the response values.

If neither the LEVELS= option nor the DISCRETE option is used, then the GMAP procedure determines the number of response levels by using the formula FLOOR(1+3.3 log(*n*)), where *n* is the number of response variable values.

By default, an equal-distribution (quantizing) algorithm is used to determine each level.

The LEVELS=*number-of-response-levels* option is ignored when you use the DISCRETE or MIDPOINTS=*value-list* option. It is also ignored when the response variables are character. When MIDPOINTS=OLD is used with the LEVELS= option, default midpoints are generated using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

MIDPOINTS=*value-list* | OLD
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 value 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. An example is:

```
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. With numeric response values, DEVICE=JAVA uses only midpoints that fall in the range of the data being used. Thus, if your data ranged from 30–80, but midpoints were specified at 25, 50, 75, and 100, only 50 and 75 are used.

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*'> **`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. Character response values specified with the MIDPOINTS= option are not supported by DEVICE=JAVA.

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 might be excluded inadvertently if values in the list are misspelled or if the case does not match exactly.

Specifying MIDPOINTS=OLD generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

Restriction Partially supported by Java

See The RANGE option

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

NOLEGEND
suppresses the legend.

PERCENT
causes GMAP to collect all response values (or their statistic) and chart each region as a percentage of the whole. You can use the STATISTICS= option to change how the percentage is calculated—whether as a percentage of the SUM, FREQUENCY, or MEAN. If you do not use the STATISTICS= option, then STATISTICS=FIRST is assumed and the response variable of only the first observation of each region is counted. If the response variable is a text field, then STATISTIC=FREQUENCY is used, even if you specify a different value for the STATISTIC= option.

Alias PERCENTAGE

See The "STATFMT=*format-specification*" on page 220, and the "STATISTIC=FIRST | SUM | FREQUENCY | MEAN" on page 221.

RANGE
causes GMAP to display, in the legend, the starting value and ending value of the range around each midpoint specified with the MIDPOINTS= option (instead of displaying just the midpoints). For example, if MIDPOINTS=15 25 35, then the legend could show 10-20, 20-30, 30-40.

Restrictions The MIDPOINTS= option must be specified for the RANGE option to have any effect.

Not supported by ActiveX.

STATFMT=*format-specification*
overrides the GMAP default format for percent of PERCENT8.2. Use this format when using calculated values. The STATFMT option is typically used when the STATISTIC=FREQUENCY option or the PERCENT option is used.

Alias SFMT=, SFORMAT=, STATFORMAT=

STATISTIC=FIRST | SUM | FREQUENCY | MEAN
specifies the statistic for GMAP to chart. For nonnumeric variables, FREQUENCY is the only allowed value—any other value is changed to FREQUENCY and a warning is issued. The frequency of a variable does not include missing values unless the MISSING option is specified.

FIRST
GMAP matches the first observation from the DATA= data set and charts the response value from this observation only. This is the default. If more rows exist that are not processed, a warning is issued to the log.

SUM
All observations matching a given ID value are added together and the summed value is charted.

FREQUENCY
A count of all rows with nonmissing values is charted unless you specify the MISSING option.

MEAN
All observations matching a given ID value are added together and then divided by the number of nonmissing observations matched. This value is then charted unless you specify the MISSING option.

Alias STAT=

UNIFORM
causes the same legend and coloring to be used for all maps produced by the procedure instead of being calculated within each BY group for each map. The UNIFORM option prescans the data to generate a categorization across all the data, regardless of BY grouping, and applies that categorization to all maps in the BY group. This results in all maps uniformly displaying legends and coloring such that a single value always has the same color in multiple maps.

When specified in a PROC GMAP statement, the UNIFORM option applies to all AREA, BLOCK, CHORO, and PRISM statements included within the GMAP run-group.

When omitted from the PROC GMAP statement, and specified on an individual AREA, BLOCK, CHORO, or PRISM statement, the UNIFORM option applies only to the maps produced by that statement.

Restriction Not supported by Java.

Details

Description

In the case of BLOCK: whereas the BLOCK statement controls the color and appearance of the blocks, the AREA statement controls the color and appearance of the regions under the block.

In the case of PRISM: whereas the PRISM statement controls the height of the prism, the AREA statement controls the color of the region. If you specify an AREA statement, the PRISM statement controls both the color and height.

BLOCK Statement

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

Requirement: 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,

Syntax

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

Summary of Optional Arguments

Appearance options

ANNOTATE=*Annotate-data-set*
: specifies a data set to annotate onto maps that are produced by the BLOCK statement.

BLOCKSIZE=*size*
: specifies the width of the blocks.

CBLKOUT=*block-outline-color* | SAME
: outlines all blocks in the specified color.

CDEFAULT=*empty-area-fill-color*
: fills empty map areas in the specified color.

CEMPTY=*empty-area-outline-color*
: outlines empty map areas in the specified color.

CEMPTY=*empty-area-outline-color*
: outlines empty map areas in the specified color.

COUTLINE=*area-outline-color* | SAME
: outlines non-empty map areas in the specified color.

COUTLINE=*area-outline-color* | SAME
: outlines non-empty map areas in the specified color.

SHAPE=*3D-block-shape*
: specifies the shape of the blocks.

STRETCH
: stretches map extents to cover all available space in the device.

UNIFORM
: causes the same legend and coloring to be used for all maps produced by the procedure instead of being calculated within each BY group for each map.

WOUTLINE=*block-outline-width*
: specifies the width, in pixels, of the outline for all outlined blocks and for the outline of the map areas.

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

YSIZE=*map-height* <*units*>
: specify the physical dimensions of the map to be drawn.

XVIEW=*x*

YVIEW=*y*

ZVIEW=*z*

specify coordinates of the viewing position in the reference coordinate system.

Description options

DESCRIPTION="*description*"

specifies a description of the output.

NAME="*name*"

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Legend options

CTEXT=*text-color*

specifies a color for the text in the legend.

LEGEND=LEGEND<1 …99>

specifies the LEGEND statement to associate with the map.

NOLEGEND

suppresses the legend.

Mapping options

AREA=*n* | *column-name*

specifies that a different map pattern be used for the surface of each map area or group of map areas on the map.

DISCRETE

generates a separate response level (color and surface pattern) for each different value of the formatted response variable.

LEVELS=*number-of-response-levels* | ALL

specifies the number of response levels to be graphed for the response variable.

MIDPOINTS=*value-list* | OLD

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

MISSING

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

PERCENT

causes GMAP to collect all response values (or their statistic) and chart each region as a percentage of the whole.

RANGE

displays value ranges in the legend.

RELZERO

creates bars and regions that are relative to a zero value.

STATFMT=*format-specification*

overrides the GMAP default format for percent of PERCENT8.2.

STATISTIC=FIRST | SUM | FREQUENCY | MEAN

specifies the statistic for GMAP to chart.

ODS options

HTML_LEGEND=*variable*

identifies the variable in the input data set whose values create links or data tips or both.

HTML=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

URL=*character-variable*
specifies a character variable whose values are URLs.

Required Argument

response-variable(s)
specifies one or more variables in the response data set. Each response variable produces a separate map. All variables must be in the input data set. Multiple response variables are separated with blanks. Blocks are not drawn for the response variable with missing values unless you use the MISSING option in the BLOCK statement.

See "About Response Variables" on page 208.

Optional Arguments

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*
specifies a data set to annotate onto maps that are produced by the BLOCK statement. Annotate coordinate systems 1, 2, 7, and 8 are not valid with block maps.

Alias ANNO=

See "Using Annotate Data Sets" in *SAS/GRAPH: Reference*.

AREA=*n* | *column-name*
specifies that a different map pattern be used for the surface of each map area or group of map areas on the map.

You can specify pattern fills or colors or both with PATTERN statements that specify map and plot patterns. A separate PATTERN definition is needed for each specified area.

AREA=*n*
The value of *n* indicates which variable in the ID statement determines the groups that are distinguished by a surface pattern. By default, all map unit areas are drawn using the same surface fill pattern. If your ID statement has only one map area identification variable, then 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, then use *n* to indicate the position of the variable that defines groups that share a pattern. When you use the AREA= option, the map data set should be sorted in order of the variables in the ID statement.

AREA=*column-name*
A column name defined in either the MAP= or DATA= data sets might be indicated with the *column-name* value. If the column name exists in both the MAP= and DATA= data sets, the column in the MAP= data set is used. When *column-name* is used, the areas are colored based on the AREA= value. Duplicate AREA= values might be assigned different patterns.

See "AREA Statement" on page 217, "PATTERN Statement" on page 44.

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

Alias BS=

CBLKOUT=*block-outline-color* | SAME
outlines all blocks in the specified color. The SAME value specifies that the outline color of a block, a block segment, or a legend is the same as the interior pattern color.

The default outline color is determined by the current style. If you specified the NOGSTYLE system option, then the default color is black for Java and ActiveX and the first color in the color list for all other devices.

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

Alias CBLOCK=

Restriction Partially supported by Java

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 are one color and you can distinguish between empty areas only by their size. Empty block patterns (VALUE=EMPTY in a PATTERN statement) are not supported by DEVICE=JAVA.

CDEFAULT=*empty-area-fill-color*
fills 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 is NONE, which draws the polygon empty, showing the background in the fill area of the polygon.

Alias CDEF=, DEFCLR=

Restriction Not supported by Java

See The CEMPTY option, the "ALL" on page 213 option, and "Displaying Map Areas and Response Data" on page 209

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.

Alias CE=

Restriction Not supported by Java

See The option "ALL" on page 213 and "Displaying Map Areas and Response Data" on page 209.

COUTLINE=*area-outline-color* | SAME

outlines non-empty map areas in the specified color. When COUTLINE=*area-outline-color* and DEVICE=JAVA or ACTIVEX, both empty and nonempty map areas are outlined. The SAME value specifies that the outline color of a map area is the same as the interior pattern color.

The default outline color is determined by the current style. If you specified the NOGSTYLE system option, then the default color is black for Java and ActiveX and the first color in the color list for all other devices.

Alias CO=

Restriction Partially supported by Java

Note If you specify empty map patterns (VALUE=EMPTY in a PATTERN statement), then you should not change the outline color from the default value SAME. Otherwise, all the outlines are one color and you cannot distinguish between the empty areas. Empty block patterns (VALUE=EMPTY in a PATTERN statement) are not supported by DEVICE=JAVA.

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:

- the CTEXT= option in a GOPTIONS statement.
- the default, the text color that is specified in the current style.
- if you specify NOGSTYLE, then the default color is black for Java and ActiveX and the first color in the color list for all other devices.

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.

Alias CT=

DESCRIPTION="*description*"

specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver). See "Chart Descriptions for Web Presentations" in *SAS/GRAPH: Reference* for more information.
- the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the Explorer.
- the **Description** field of the PROC GREPLAY window.

The *description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to "Substituting BY Line Values in a Text String" on page 73. The 256-character limit applies before the substitution takes place for these options.

Thus, if in the SAS program the entry-description text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

Alias DES=

Default BLOCK MAP OF *variable-name*

DISCRETE

generates a separate response level (color and surface pattern) for each different value of the formatted response variable.

The LEVELS=*number-of-response-levels* option is ignored when you use the DISCRETE option.

If you specify the DISCRETE option, then distinct, non-continuous colors are used for the response values. If you specify the LEVELS= option, then a color ramp is used to assign each response value a continuous color scheme.

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

HTML=*variable*

identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

See "Data Tips for Web Presentations" in *SAS/GRAPH: Reference*

"Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" in *SAS/GRAPH: Reference*

HTML_LEGEND=*variable*

identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

Restriction Not supported by Java and ActiveX

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" in *SAS/GRAPH: Reference*

LEGEND=LEGEND<1 …99>

specifies the LEGEND statement to associate with the map. The LEGEND= option 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. Most of the LEGEND options described in "LEGEND Statement" on page 25 are supported by both Java and ActiveX. If a LEGEND option is not supported by Java or ActiveX, it is noted in the LEGEND option definition.

Restriction Partially supported by Java and ActiveX

See "LEGEND Statement" on page 25

LEVELS=*number-of-response-levels* | ALL
specifies the number of response levels to be graphed for the response variable.

If you specify LEVELS=ALL , then all unique numeric or character response variable values are graphed.

Each response level is assigned a different surface pattern and color combination. The block height is based on the data value of the corresponding response variable.

If you specify the LEVELS= option, then a color ramp is used to assign each response value a continuous color scheme. The response values are assigned lighter and darker values of a color scheme to express lower and higher response values. If you specify the DISCRETE option, then distinct, non-continuous colors are used for the response values.

If neither the LEVELS= option nor the DISCRETE option is used, then the GMAP procedure determines the number of response levels by using the formula FLOOR(1+3.3 log(*n*)), where *n* is the number of response variable values.

By default, an equal-distribution (quantizing) algorithm is used to determine each level.

The LEVELS=*number-of-response-levels* option is ignored when you use the DISCRETE or MIDPOINTS=*value-list* option. It is also ignored when the response variables are character. When MIDPOINTS=OLD is used with the LEVELS= option, default midpoints are generated using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

Note If you specified the NOGSTYLE system option, then noncontinuous colors are used by default.

MIDPOINTS=*value-list* | OLD
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 value 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. With numeric response values, DEVICE=JAVA uses only midpoints that fall in the range of the data being used. Thus, if your data ranged from 30–80, but midpoints were specified at 25, 50, 75, and 100, only 50 and 75 are used.

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*'> **`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. Character response values specified with the MIDPOINTS= option are not supported by DEVICE=JAVA.

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 might be excluded inadvertently if values in the list are misspelled or if the case does not match exactly.

Specifying MIDPOINTS=OLD generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

Restriction Partially supported by Java

See The RANGE option

Example "Example 17: Using GfK GeoMarketing Map Data When Specifying Midpoints in a Prism Map" on page 320

MISSING

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

See "Displaying Map Areas and Response Data" on page 209.

NAME="*name*"

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.
- **For the graphics output filename:**
  - The filename is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output filename. In this case, the file reference specification overrides the NAME= value. See "Controlling Graphics Output for ODS LISTING" in *SAS/GRAPH: Reference*.
  - All characters are represented in lowercase.
  - If a number is added to the GRSEG name, the same number is added to the output filename. See "About Filename Indexing" in *SAS/GRAPH: Reference*.

- If the NAME= value is 8 characters or less, the filename is the GRSEG entry name.
- If the NAME= value is greater than 8 characters, the NAME= value is used as the filename. When an index number is used in the GRSEG entry name, that index number is appended to the output filename. See "About Filename Indexing" in *SAS/GRAPH: Reference*.

Note: If the name begins with a number, an underscore is prepended to the filename.

- The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default GMAP

NOLEGEND
suppresses the legend.

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.

Alias CE=

Restriction Not supported by Java

See The option "ALL" on page 213 and "Displaying Map Areas and Response Data" on page 209.

COUTLINE=*area-outline-color* | SAME
outlines non-empty map areas in the specified color. When COUTLINE=*area-outline-color* and DEVICE=JAVA or ACTIVEX, both empty and nonempty map areas are outlined. The SAME value specifies that the outline color of a map area is the same as the interior pattern color.

The default outline color is determined by the current style. If you specified the NOGSTYLE system option, then the default color is black for Java and ActiveX and the first color in the color list for all other devices.

Alias CO=

Restriction Partially supported by Java

Note If you specify empty map patterns (VALUE=EMPTY in a PATTERN statement), then you should not change the outline color from the default value SAME. Otherwise, all the outlines are one color and you cannot distinguish between the empty areas. Empty block patterns (VALUE=EMPTY in a PATTERN statement) are not supported by DEVICE=JAVA.

PERCENT
causes GMAP to collect all response values (or their statistic) and chart each region as a percentage of the whole. You can use the STATISTIC= option to change how the percentage is calculated—whether as a percentage of the SUM, FREQUENCY, or

MEAN. If you do not use the STATISTIC= option, then STATISTIC=FIRST is assumed and the response variable of only the first observation of each region is counted. If the response variable is a text field, then STATISTIC=FREQUENCY is used, even if you specify a different value for the STATISTIC= option.

Alias PERCENTAGE

See The "STATFMT=*format-specification*" on page 231, and the "STATISTIC=FIRST | SUM | FREQUENCY | MEAN" on page 232

RANGE

causes GMAP to display, in the legend, the starting value and ending value of the range around each midpoint specified with the MIDPOINTS= option (instead of displaying just the midpoints). For example, if MIDPOINTS=15 25 35, then the legend could show 10-20, 20-30, 30-40.

Restriction MIDPOINTS= must be specified for the RANGE option to have any effect. Not supported by ActiveX.

RELZERO

creates bars and regions that are relative to a zero value. By default, GMAP creates heights that are relative to the minimum value, which might or might not be zero. With the RELZERO option, zero value bars have no height.

Alias REL0, RELATIVETOZERO

Restrictions This option works only for variables that have no negative values.

Not supported by Java

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. Unless you specify SHAPE=OLD, only solid fill patterns are used. The value of *3D-block-shape* can be one of the following:

- BLOCK | B
- CYLINDER | C
- HEXAGON | H
- OLDBLOCK | OLD
- PRISM | P
- STAR | S

SHAPE=BLOCK is the default. OLDBLOCK is the same as BLOCK except that with OLDBLOCK the tops and sides of blocks are colored the same as the background. This exception existed prior to the SAS 9.2 release.

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

Default BLOCK

STATFMT=*format-specification*

overrides the GMAP default format for percent of PERCENT8.2. Use this format when using calculated values. The STATFMT option is typically used when the STATISTIC=FREQUENCY option or the PERCENT option is used.

Alias SFMT=, SFORMAT=, STATFORMAT=

STATISTIC=FIRST | SUM | FREQUENCY | MEAN

specifies the statistic for GMAP to chart. For character variables, FREQUENCY is the only allowed value—any other value is changed to FREQUENCY and a warning is issued. The frequency of a variable does not include missing values unless the MISSING option is specified.

FIRST

GMAP matches the first observation from the DATA= data set and charts the response value from this observation only. This is the default. If more rows exist that are not processed, a warning is issued to the log.

SUM

All observations matching a given ID value are added together and the summed value is charted.

FREQUENCY

A count of all rows with nonmissing values is charted unless you specify the MISSING option.

MEAN

All observations matching a given ID value are added together and then divided by the number of nonmissing observations matched. This value is then charted unless you specify the MISSING option.

Alias STAT=

Example "Example 7: Using GfK GeoMarketing Map Data When Specifying the Statistic for the Response Variable" on page 297

STRETCH

stretches map extents to cover all available space in the device. This might cause the map to be distorted. When this option is applied to the PROC GMAP statement, it applies to all statements. If applied to a single statement, it applies only to that statement.

Alias STRETCHTOFIT, STR2FIT

Restriction Not supported by Java and ActiveX

UNIFORM

causes the same legend and coloring to be used for all maps produced by the procedure instead of being calculated within each BY group for each map. The UNIFORM option prescans the data to generate a categorization across all the data, regardless of BY grouping, and applies that categorization to all maps in the BY group. This results in all maps uniformly displaying legends and coloring such that a single value always has the same color in multiple maps.

When specified in a PROC GMAP statement, the UNIFORM option applies to all AREA, BLOCK, CHORO, and PRISM statements included within the GMAP run-group.

When omitted from the PROC GMAP statement, and specified on an individual AREA, BLOCK, CHORO, or PRISM statement, the UNIFORM option applies only to the maps produced by that statement.

Restriction Not supported by Java

URL=*character-variable*

specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restriction This option affects graphics output that is created through the ODS HTML destination only.

Interaction If you specify both the HTML= option and URL= option, then the URL= option is ignored.

See "Overview of Enhancing Web Presentations" in *SAS/GRAPH: Reference*

"Example: GIF Output with Drill-Down Links" in *SAS/GRAPH: Reference*

WOUTLINE=*block-outline-width*
specifies the width, in pixels, of the outline for all outlined blocks and for the outline of the map areas.

Default 1

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

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

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

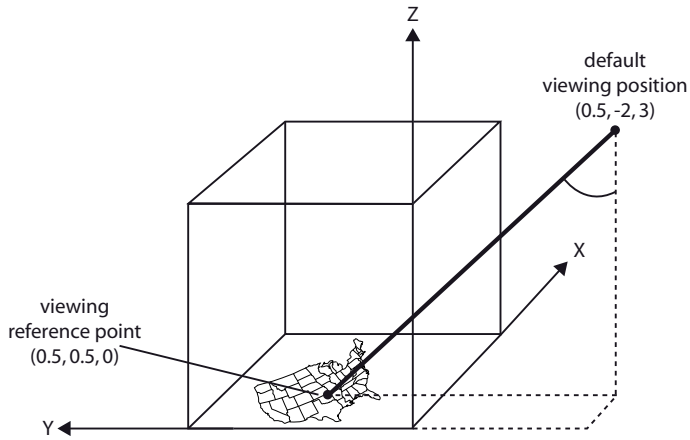
Restriction Not supported by Java and ActiveX

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. 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. You can specify one, two, or all three of the view coordinates; any that you do not specify are assigned the default values. Although you can use the XVIEW= and YVIEW= options with DEVICE=JAVA, ZVIEW= cannot be used with DEVICE=JAVA.

Figure 7.5 on page 234 shows the position of the viewing reference point, as well as the default viewing position.

Figure 7.5 *Viewing Position and Viewing Reference Point*



Alias	XV=, YV=, ZV=
Restriction	Partially supported by Java

Details

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 performs the following operations:

- determines the midpoints ranges.
- scales the blocks.
- assigns patterns to the block faces and map areas. (See "About Block Maps and Patterns" on page 234 for more information.)

You can use statement options to enhance the appearance of the map. For example, you can specify the width and shape 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.

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, block patterns are determined by the current style. When specifying the AREA statement or the AREA= option, the map area colors are determined by the current style and the block colors are determined by the specified attributes.

Note: If you specified the NOGSTYLE system option, then solid patterns are used for blocks and hatch patterns are used for the map areas. The map areas and their outlines use the first color in the color list.

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 and AREA statement control how the map areas are patterned.

When you use PATTERN statements to define the patterns for the map, you must specify the correct type of pattern for the area. The blocks use bar and block patterns and the map areas use map and plot patterns. See "PATTERN Statement" on page 44 for more information about specifying patterns.

Note: If you specify only one PATTERN statement and include only the COLOR= option, that color is 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;
```

Note: Empty block patterns (VALUE=EMPTY in a PATTERN statement) are not supported by DEVICE=JAVA.

Note: If you specify a PATTERN statement to define a non-solid fill pattern but do not specify the BLOCK statement's SHAPE=OLD option, the blocks are filled with solid patterns.

CHORO Statement

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

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

Global statements: FOOTNOTE , NOTE, LEGEND, PATTERN, TITLE

Syntax

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

Summary of Optional Arguments

Appearance options

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

CDEFAULT=*empty-area-fill-color*
: fills empty map areas in the specified color.

CEMPTY=*empty-area-outline-color*
: outlines empty map areas in the specified color.

COUTLINE=*area-outline-color* | SAME
: outlines non-empty map areas in the specified color.

OSM <=(<STYLE=osmstyle> <AUTOPROJECT>)>
: specifies an OpenStreetMap (OSM) style and can project the unprojected map data from latitude and longitude degrees onto the OSM map.

STRETCH
: stretches map extents to cover all available space in the device.

UNIFORM
: causes the same legend and coloring to be used for all maps produced by the procedure instead of being calculated within each BY group for each map.

WOUTLINE=*area-outline-width*
: specifies the width of all map area outlines, in pixels.

XSIZE=*map-width <units>*

YSIZE=*map-height <units>*
: specify the physical dimensions of the map.

Description options

DESCRIPTION="*description*"
: specifies a description of the output.

NAME="*name*"
: specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Legend options

CTEXT=*text-color*
: specifies a color for the text in the legend.

LEGEND=LEGEND<1 …99>
: assigns the specified LEGEND statement that is to be applied to the map.

NOLEGEND
: suppresses the legend.

Mapping options

DISCRETE
: generates a separate response level (color and surface pattern) for each different value of the formatted response variable.

LEVELS=*number-of-response-levels* | ALL
: specifies the number of response levels to be graphed for the response variable.

MIDPOINTS=*value-list* | OLD
: specifies the response levels for the range of response values that are represented by each level (pattern and color combination).

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

PERCENT
: causes GMAP to collect all response values (or their statistic) and chart each region as a percentage of the whole.

RANGE
: displays value ranges in the legend.

STATFMT=*format-specification*
: overrides the GMAP default format for percent of PERCENT8.2.

STATISTIC=FIRST | SUM | FREQUENCY | MEAN
: specifies the statistic for GMAP to chart.

ODS options

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

HTML=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

URL=*character-variable*
specifies a character variable whose values are URLs.

Required Argument

response-variable(s)
specifies one or more variables in the response data set. Each response variable produces a separate map. All variables must be in the input data set. Multiple response variables are separated with blanks.

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

Response variables can be either numeric or character in type. Numeric response variables are normally grouped into ranges, or response levels, as determined by default, or by the MIDPOINTS= or LEVELS=*number-of-response-levels* options. Each response level is assigned a different combination of pattern and color. With the LEVELS=ALL option, numeric or character response variables are assigned unique response levels, as are numeric variables when the DISCRETE option is specified. The LEVELS=*number-of-response-levels* option is ignored when either the DISCRETE or MIDPOINTS= option is used.

See "About Response Variables" on page 208.

Optional Arguments

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*
specifies a data set to annotate onto maps that are produced by the CHORO statement.

Alias ANNO=

See "Using Annotate Data Sets" in *SAS/GRAPH: Reference*

Example "Example 12: Using GfK GeoMarketing Map Data When Labeling Provinces on a Map" on page 310

CDEFAULT=*empty-area-fill-color*
fills empty map areas in the specified color. This option affects only map areas that are empty. Empty map areas are generated in choro maps only when there is no response value for a map area and the MISSING option is not used. They are also generated when a map area is omitted from the response data set and the ALL option is included in the PROC GMAP statement.

The default is NONE, which draws the polygon empty, showing the background in the fill area of the polygon.

Alias CDEF=, DEFCLR=

Restriction Not supported by Java

See The CEMPTY option, the "ALL" on page 213, and "Displaying Map Areas and Response Data" on page 209

CEMPTY=*empty-area-outline-color*
outlines empty map areas in the specified color. This option affects only the empty map areas, which are generated in choro maps when either of the following is true:

- There is no response value for a map area and the MISSING option is not used.
- 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.

Alias CE=

Restriction Not supported by Java

See The option "ALL" on page 213 and "Displaying Map Areas and Response Data" on page 209.

COUTLINE=*area-outline-color* | SAME
outlines non-empty map areas in the specified color. When COUTLINE=*area-outline-color* and DEVICE=JAVA or ACTIVEX, both empty and non-empty map areas are outlined. The value SAME specifies that the outline color of a map area is the same as the interior pattern color.

The default outline color is determined by the current style. If you specified the NOGSTYLE system option, then the default color is black for Java and ActiveX and the first color in the color list for all other devices.

Alias CO=

Note If you specify empty map patterns (VALUE=EMPTY in a PATTERN statement), then you should not change the outline color from the default value SAME to a single color. Otherwise, all the outlines are one color and you cannot distinguish between the empty areas.

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:

- the CTEXT= option in a GOPTIONS statement.
- the default, the text color that is specified in the current style.
- If you specified the NOGSTYLE system option, then the default color is black for Java and ActiveX and the first color in the color list for all other devices.

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.

Alias CT=

DESCRIPTION="*description*"

specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver). See "Chart Descriptions for Web Presentations" in *SAS/GRAPH: Reference* for more information.
- the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the Explorer.
- the **Description** field of the PROC GREPLAY window.

The *description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to "Substituting BY Line Values in a Text String" on page 73. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the entry-description text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

Alias DES=

Default CHOROPLETH MAP OF *variable-name*

DISCRETE

generates a separate response level (color and surface pattern) for each different value of the formatted response variable.

The LEVELS=*number-of-response-levels* option is ignored when you use the DISCRETE option.

If you specify the DISCRETE option, then distinct, non-continuous colors are used for the response values. If you specify the LEVELS= option, then a color ramp is used to assign each response value a continuous color scheme.

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

HTML=*variable*

identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

See "Data Tips for Web Presentations" in *SAS/GRAPH: Reference*

"Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" in *SAS/GRAPH: Reference*

HTML_LEGEND=*variable*

identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated

with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend.

Restriction Not supported by Java and ActiveX

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" in *SAS/GRAPH: Reference*

LEGEND=LEGEND<1 …99>

assigns the specified LEGEND statement that is to be applied to the map. The LEGEND= option is ignored if the specified LEGEND definition is not currently in effect. In the GMAP procedure, the CHORO statement produces a legend by default unless you specify the NOLEGEND option. If you use the SHAPE= option in a LEGEND statement, then only the value BAR is valid. Most of the LEGEND options described in "LEGEND Statement" on page 25 are supported by both Java and ActiveX. If a LEGEND option is not supported by Java or ActiveX, it is noted in the LEGEND option definition.

Restriction Partially supported by Java and ActiveX

See "LEGEND Statement" on page 25

Example "Example 5: Using GfK GeoMarketing Map Data to Assign a Format to the Response Variable" on page 290

LEVELS=*number-of-response-levels* | ALL

specifies the number of response levels to be graphed for the response variable.

If you specify LEVELS=ALL , then all unique numeric or character response variable values are graphed.

Each response level is assigned a different surface pattern and color combination.

If you specify the LEVELS= option, then a color ramp is used to assign each response value a continuous color scheme. The response values are assigned lighter and darker values of a color scheme to express lower and higher response values. If you specify the DISCRETE option, then distinct, non-continuous colors are used for the response values.

If neither the LEVELS= option nor the DISCRETE option is used, then the GMAP procedure determines the number of response levels by using the formula FLOOR(1+3.3 log(n)), where n is the number of response variable values.

By default, an equal-distribution (quantizing) algorithm is used to determine each level.

When MIDPOINTS=OLD is used with the LEVELS= option, default midpoints are generated using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

Restriction The LEVELS=*number-of-response-levels* option is ignored when you use the DISCRETE or MIDPOINTS=*value-list* option. It is also ignored when the response variables are character.

Note If you specified the NOGSTYLE system option, then noncontinuous colors are used by default.

Example "Example 3: Using GfK GeoMarketing Map Data to Specify Response Levels in a Block Map" on page 286

MIDPOINTS=*value-list* | OLD

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

For numeric response variables, the *value-list* argument is either an explicit list of values, a starting value 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. With numeric response values, DEVICE=JAVA uses only midpoints that fall in the range of the data being used. Thus, if your data ranged from 30–80, but midpoints were specified at 25, 50, 75,and 100, only 50 and 75 are used.

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. Character response values specified with the MIDPOINTS= option are not supported by DEVICE=JAVA.

You can selectively exclude some response variable values from the map, as shown here: **`midpoints="Midwest"`**

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

Specifying MIDPOINTS=OLD generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

Restriction Partially supported by Java

See The RANGE option

Example "Example 17: Using GfK GeoMarketing Map Data When Specifying Midpoints in a Prism Map" on page 320

MISSING

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

See "Displaying Map Areas and Response Data" on page 209

NAME="*name*"

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.
- **For the graphics output filename:**
  - The filename is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output filename. In this case, the file reference specification overrides the NAME= value. See "Controlling Graphics Output for ODS LISTING" in *SAS/GRAPH: Reference*.
  - All characters are represented in lowercase.
  - If a number is added to the GRSEG name, the same number is added to the output filename. See "About Filename Indexing" in *SAS/GRAPH: Reference*.
  - If the NAME= value is 8 characters or less, the filename is the GRSEG entry name.
  - If the NAME= value is greater than 8 characters, the NAME= value is used as the filename. When an index number is used in the GRSEG entry name, that index number is appended to the output filename. See "About Filename Indexing" in *SAS/GRAPH: Reference*.

  Note: If the name begins with a number, an underscore is prepended to the filename.
  - The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default GMAP

NOLEGEND

suppresses the legend.

Example "Example 12: Using GfK GeoMarketing Map Data When Labeling Provinces on a Map" on page 310

OSM <=(<STYLE=osmstyle> <AUTOPROJECT>)>

specifies an OpenStreetMap (OSM) style and can project the unprojected map data from latitude and longitude degrees onto the OSM map. This is an appearance option that enables you to use the OpenStreetMap (OSM) map as a background map. You can specify no suboptions, or use either a STYLE= suboption or an AUTOPROJECT suboption, or both suboptions.

If you specify the OSM option without any suboptions, the GMAP procedure by default uses the SASMAPNIK style and does not project the map data.

If you specify the STYLE=osmstyle suboption only, the GMAP procedure uses one of the supported OSM styles that are appropriate for the map that you are processing (for example, SASMAPNIK or SASMAPNIK_LITE). Because the AUTOPROJECT suboption is not specified, the GMAP procedure does not project the map data.

If you specify the OSM option with the AUTOPROJECT suboption only, the GMAP procedure projects the map data from latitude and longitude coordinates (in degrees) onto the OSM map. Because the STYLE suboption is not specified, the procedure uses the default SASMAPNIK style.

If you specify the OSM option with both the STYLE= and AUTOPROJECT suboptions, then the GMAP procedure uses one of the supported OSM styles that are appropriate for the map that you are processing. It projects the map data from latitude and longitude coordinates (in degrees) onto the OSM map.

Alias SHOWOSM

Default The default STYLE is SASMAPNIK. The map data is not projected unless you specify the AUTOPROJECT suboption.

Restrictions The OSM option supports only JAVA and JAVAIMG devices.

If the map data is already projected, do not specify the AUTOPROJECT suboption, or you will receive unexpected results.

PERCENT

causes GMAP to collect all response values (or their statistic) and chart each region as a percentage of the whole. You can use the STATISTIC= option to change how the percentage is calculated—whether as a percentage of the SUM, FREQUENCY, or MEAN. If you do not use the STATISTIC= option, then STATISTIC=FIRST is assumed–the response variable of only the first observation of each region is counted. If the response variable is a text field, then STATISTIC=FREQUENCY is used, even if you specify a different value for the STATISTIC= option.

Alias PERCENTAGE

See The "STATFMT=*format-specification*" on page 243 and the "STATISTIC=FIRST | SUM | FREQUENCY | MEAN" on page 244.

RANGE

causes GMAP to display, in the legend, the starting value and ending value of the range around each midpoint specified with the MIDPOINTS= option (instead of displaying just the midpoints). For example, if MIDPOINTS=15 25 35, then the legend could show 10-20, 20-30, 30-40.

Restrictions The MIDPOINTS= option must be specified for the RANGE option to have any effect.

Not supported by ActiveX

STATFMT=*format-specification*

overrides the GMAP default format for percent of PERCENT8.2. Use this format when using calculated values. The STATFMT option is typically used when the STATISTIC=FREQUENCY option or the PERCENT option is used.

Alias SFMT=, SFORMAT=, STATFORMAT=

STATISTIC=FIRST | SUM | FREQUENCY | MEAN

specifies the statistic for GMAP to chart. For character variables, FREQUENCY is the only allowed value—any other value is changed to FREQUENCY and a warning is issued. The frequency of a variable does not include missing values unless the MISSING option is specified.

FIRST

GMAP matches the first observation from the DATA= data set and charts the response value from this observation only. This is the default. If more rows exist that are not processed, a warning is issued to the log.

SUM

All observations matching a given ID value are added together and the summed value is charted.

FREQUENCY

A count of all rows with nonmissing values is charted unless you specify the MISSING option.

MEAN

All observations matching a given ID value are added together and then divided by the number of nonmissing observations matched. This value is then charted unless you specify the MISSING option.

Alias STAT=

STRETCH

stretches map extents to cover all available space in the device. This might cause the map to be distorted. When this option is applied to the PROC GMAP statement, it applies to all statements. If applied to a single statement, it applies only to that statement.

Alias STRETCHTOFIT, STR2FIT

Restriction Not supported by Java and ActiveX

UNIFORM

causes the same legend and coloring to be used for all maps produced by the procedure instead of being calculated within each BY group for each map. The UNIFORM option prescans the data to generate a categorization across all the data, regardless of BY grouping, and applies that categorization to all maps in the BY group. This results in all maps uniformly displaying legends and coloring such that a single value always has the same color in multiple maps.

When specified in a PROC GMAP statement, UNIFORM applies to all AREA, BLOCK, CHORO, and PRISM statements included within the GMAP run-group.

When omitted from the PROC GMAP statement, and specified on an individual AREA, BLOCK, CHORO, or PRISM statement, UNIFORM applies only to the maps produced by that statement.

Restriction Not supported by Java

URL=*character-variable*

specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restriction This option affects graphics output that is created through the ODS HTML destination only.

Interaction If you specify both the HTML= and URL= options, then the URL= option is ignored

See "Overview of Enhancing Web Presentations" in *SAS/GRAPH: Reference*

"Example: GIF Output with Drill-Down Links" in *SAS/GRAPH: Reference*

WOUTLINE=*area-outline-width*
specifies the width of all map area outlines, in pixels.

Default 1

XSIZE=*map-width <units>*
YSIZE=*map-height <units>*
specify the physical dimensions of the map. By default, the map uses the entire procedure output area.

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

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

You can specify only one of the XSIZE= or YSIZE= options. In this case the GMAP procedure scales the dimension for the unspecified option in order to retain the original shape of the map.

Restriction Not supported by Java and ActiveX

Details

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.

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.

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

Global statements: FOOTNOTE , NOTE, LEGEND, PATTERN, TITLE

Syntax

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

Summary of Optional Arguments

Appearance options

ANNOTATE=*Annotate-data-set*
: specifies a data set to annotate onto the maps that are produced by the PRISM statement.

CDEFAULT=*empty-area-fill-color*
: fills empty map areas in the specified color.

CDEFAULT=*empty-area-fill-color*
: fills empty map areas in the specified color.

CEMPTY=*empty-area-outline-color*
: outlines empty map areas in the specified color.

COUTLINE=*area-outline-color* | SAME
: outlines nonempty map areas in the specified color.

STRETCH
: stretches map extents to cover all available space in the device.

UNIFORM
: causes the same legend and coloring to be used for all maps produced by the procedure instead of being calculated within each BY group for each map.

WOUTLINE=*area-outline-width*
: specifies the width, in pixels, of all map area outlines.

XLIGHT=*x*

YLIGHT=*y*
: specify the coordinates of the imaginary light source in the map coordinate system.

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

YSIZE=*map-height* <*units*>
: specify the dimensions of the map that you are drawing.

XVIEW=*x*

YVIEW=*y*

ZVIEW=*z*
: specify the viewing position coordinates for the map.

Description options

DESCRIPTION="*description*"
: specifies a description of the output.

NAME="*name*"
: specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Legend options

CTEXT=*text-color*

LEGEND=LEGEND<1 …99>
specifies the LEGEND definition to associate with the map.

NOLEGEND
suppresses the legend.

Mapping options

AREA=*n* | *column-name*
specifies that a different map pattern be used for the surface of each map area or group of map areas on the map.

DISCRETE
generates a separate response level (color and surface pattern) for each different value of the formatted response variable.

LEVELS=*number-of-response-levels* | ALL
specifies the number of response levels to be graphed for the response variable.

MIDPOINTS=*value-list* | OLD
specifies the response levels for the range of response values that are represented by each level (prism height, pattern, and color combination).

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

PERCENT
causes GMAP to collect all response values (or their statistic) and chart each region as a percentage of the whole.

RANGE

RELZERO
creates area heights that are relative to a zero value.

STATFMT=*format-specification*
overrides the GMAP default format for percent of PERCENT8.2.

STATISTIC=FIRST | SUM | FREQUENCY | MEAN
specifies the statistic for GMAP to chart.

ODS options

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

HTML=*variable*
identifies the variable in the input data set whose values create links or data tips or both.

URL=*character-variable*
specifies a character variable whose values are URLs.

Required Argument

response-variable(s)
specifies one or more variables in the response data set. Each response variable produces a separate map. All variables must be in the input data set. Multiple response variables are separated with blanks.

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

Response variables can be either numeric or character. By default, and as determined by the LEVELS= or MIDPOINTS= values, numeric response variables are grouped into ranges, or response levels. Each response level is assigned a different prism height and a different pattern and color combination. With the LEVELS=ALL option, numeric or character response variables are assigned unique response levels, as are numeric variables when the DISCRETE option is specified. The LEVELS=*number-of-response-levels* option is ignored when either the DISCRETE or MIDPOINTS= option is used.

See "About Response Variables" on page 208.

Optional Arguments

Options in a PRISM statement affect all of the 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*

specifies a data set to annotate onto the maps that are produced by the PRISM statement. Annotate coordinate systems 1, 2, 7, and 8 are not valid with Prism maps.

Alias ANNO=

See "Using Annotate Data Sets" in *SAS/GRAPH: Reference*

AREA=*n* | *column-name*

specifies that a different map pattern be used for the surface of each map area or group of map areas on the map.

You can specify pattern fills or colors or both with PATTERN statements that specify map and plot patterns. A separate PATTERN definition is needed for each specified area.

AREA=*n*

The value of *n* indicates which variable in the ID statement determines the groups that are distinguished by a surface pattern. By default, all map unit areas are drawn using the same surface fill pattern. If your ID statement has only one map area identification variable, then 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, then use *n* to indicate the position of the variable that defines groups that share a pattern. When you use the AREA= option, the map data set should be sorted in order of the variables in the ID statement.

AREA=*column-name*

A column name defined in either the MAP= or DATA= data sets can be indicated with the *column-name* value. If the column name exists in both the MAP= and DATA= data sets, the column in the map= data set is used. When *column-name* is used, the areas are colored based on the AREA= value. Duplicate AREA= values might be assigned different patterns.

Note The AREA statement provides a greater amount of control than the AREA= option.

See "AREA Statement" on page 217, "PATTERN Statement" on page 44.

CDEFAULT=*empty-area-fill-color*

fills empty map areas in the specified color. This option affects only map areas that are empty. Empty map areas are generated in prism maps only when there is no response value for a map area and the MISSING option is not used. They are also

generated when a map area is omitted from the response data set and the ALL option is included in the PROC GMAP statement.

The default is NONE, which draws the polygon empty, showing the background in the fill area of the polygon.

Alias CDEF=, DEFCLR=

Restriction Not supported by Java

See The CEMPTY option, the "ALL" on page 213, and "Displaying Map Areas and Response Data" on page 209

CEMPTY=*empty-area-outline-color*
outlines empty map areas in the specified color. 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.

Alias CE=

Restriction Not supported by Java

See "ALL" on page 213 and "Displaying Map Areas and Response Data" on page 209

CDEFAULT=*empty-area-fill-color*
fills empty map areas in the specified color. This option affects only map areas that are empty. Empty map areas are generated in prism maps only when there is no response value for a map area and the MISSING option is not used. They are also generated when a map area is omitted from the response data set and the ALL option is included in the PROC GMAP statement.

The default is NONE, which draws the polygon empty, showing the background in the fill area of the polygon.

Alias CDEF=, DEFCLR=

Restriction Not supported by Java

See The CEMPTY option, the "ALL" on page 213, and "Displaying Map Areas and Response Data" on page 209

COUTLINE=*area-outline-color* | SAME
outlines nonempty 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 is determined by the current style. If you specified the NOGSTYLE system option, then the default color is the first color in the color list.

Alias CO=

Note If you specify empty map 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 are one color and you

cannot distinguish between the empty areas. Empty block patterns (VALUE=EMPTY in a PATTERN statement) are not supported by DEVICE=JAVA.

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:

- the CTEXT= option in a GOPTIONS statement.
- the default, the text color that is specified in the current style.
- If you specified the NOGSTYLE system option, then the default color is black for Java and ActiveX and the first color in the color list for all other devices.

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.

Alias CT=

DESCRIPTION="*description*"
specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver). See "Chart Descriptions for Web Presentations" in *SAS/GRAPH: Reference* for more information.
- the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the Explorer.
- the **Description** field of the PROC GREPLAY window.

The *description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to "Substituting BY Line Values in a Text String" on page 73. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the entry-description text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

Alias DES=

Default PRISM MAP OF *variable-name*

DISCRETE
generates a separate response level (color and surface pattern) for each different value of the formatted response variable.

The LEVELS=*number-of-response-levels* option is ignored when you use the DISCRETE option.

If you specify the DISCRETE option, then distinct, non-continuous colors are used for the response values. If you specify the LEVELS= option, then a color ramp is used to assign each response value a continuous color scheme.

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

HTML=*variable*
identifies the variable in the input data set whose values create links or data tips or both. The variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are URLs pointing to web pages to display when the user clicks (drills down) on elements in the graph. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over elements in the graph.

See "Data Tips for Web Presentations" in *SAS/GRAPH: Reference*

"Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" in *SAS/GRAPH: Reference*

HTML_LEGEND=*variable*
identifies the variable in the input data set whose values create links or data tips or both. Input data set variable values are either links or data tips or both that are created in the HTML file generated by the ODS statement. The links are associated with a legend value and point to the URL to display when the user clicks (drills down) on the value. Data tips are detailed information or data values that are displayed as pop-up text when a mouse pointer is positioned over values in the legend..

Restriction Not supported by Java and ActiveX

See "Adding Links and Enhancements with the URL=, HTML=, and HTML_LEGEND= Options" in *SAS/GRAPH: Reference*

LEGEND=LEGEND<1 …99>
specifies the LEGEND definition to associate with the map. 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. Most of the LEGEND options described in "LEGEND Statement" on page 25 are supported by both Java and ActiveX. If a LEGEND option is not supported by Java or ActiveX, it is noted in the LEGEND option definition.

Restriction Partially supported by Java and ActiveX

See "LEGEND Statement" on page 25

Example "Example 17: Using GfK GeoMarketing Map Data When Specifying Midpoints in a Prism Map" on page 320

LEVELS=*number-of-response-levels* | ALL
specifies the number of response levels to be graphed for the response variable.

If you specify LEVELS=ALL, then all unique numeric or character response variable values are graphed.

Each response level is assigned a different surface pattern and color combination. The prism height is based on the data value of the corresponding response variable.

If you specify the LEVELS= option, then a color ramp is used to assign each response value a continuous color scheme. The response values are assigned lighter and darker values of a color scheme to express lower and higher response values. If

you specify the DISCRETE option, then distinct, non-continuous colors are used for the response values.

If neither the LEVELS= option nor the DISCRETE option is used, then the GMAP procedure determines the number of response levels by using the formula FLOOR(1+3.3 log(*n*)), where *n* is the number of response variable values.

By default, an equal-distribution (quantizing) algorithm is used to determine each level.

When MIDPOINTS=OLD is used with the LEVELS= option, default midpoints are generated using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

Restriction The LEVELS=*number-of-response-levels* option is ignored when you use the DISCRETE or MIDPOINTS=*value-list* option. It is also ignored when the response variables are character.

Example "Example 3: Using GfK GeoMarketing Map Data to Specify Response Levels in a Block Map" on page 286

MIDPOINTS=*value-list* | OLD

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 value 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. An example is:
`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. With numeric response values, DEVICE=JAVA uses only midpoints that fall in the range of the data being used. Thus, if your data ranged from 30–80, but midpoints were specified at 25, 50, 75, and 100, only 50 and 75 are used.

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, as shown here: **`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. Character response values specified with the MIDPOINTS= option are not supported by DEVICE=JAVA.

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 might be inadvertently excluded if values in the list are misspelled or if the case does not match exactly.

Specifying MIDPOINTS=OLD generates default midpoints using the Nelder algorithm (*Applied Statistics* 25:94–7, 1976).

Restriction Partially supported by Java

See The RANGE option

Example "Example 17: Using GfK GeoMarketing Map Data When Specifying Midpoints in a Prism Map" on page 320

MISSING

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

See "Displaying Map Areas and Response Data" on page 209

NAME="*name*"

specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.
- **For the graphics output filename:**
  - The filename is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output filename. In this case, the file reference specification overrides the NAME= value. See "Controlling Graphics Output for ODS LISTING" in *SAS/GRAPH: Reference*.
  - All characters are represented in lowercase.
  - If a number is added to the GRSEG name, the same number is added to the output filename. See "About Filename Indexing" in *SAS/GRAPH: Reference*.
  - If the NAME= value is 8 characters or less, the filename is the GRSEG entry name.
  - If the NAME= value is greater than 8 characters, the NAME= value is used as the filename. When an index number is used in the GRSEG entry name, that index number is appended to the output filename. See "About Filename Indexing" in *SAS/GRAPH: Reference*.

  Note: If the name begins with a number, an underscore is prepended to the filename.

- The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default GMAP

NOLEGEND
suppresses the legend.

PERCENT
causes GMAP to collect all response values (or their statistic) and chart each region as a percentage of the whole. You can use the STATISTIC= option to change how the percentage is calculated—whether as a percentage of the SUM, FREQUENCY, or MEAN. If you do not use the STATISTIC= option, then STATISTIC=FIRST is assumed and the response variable of only the first observation of each region is counted. If the response variable is a text field, then STATISTIC=FREQUENCY is used, even if you specify a different value for the STATISTIC= option.

Alias PERCENTAGE

See The "STATFMT=*format-specification*" on page 254, and the "STATISTIC=FIRST | SUM | FREQUENCY | MEAN" on page 254.

RANGE
causes GMAP to display, in the legend, the starting value and ending value of the range around each midpoint specified with the MIDPOINTS= option (instead of displaying just the midpoints). For example, if MIDPOINTS=15 25 35, then the legend could show 10-20, 20-30, 30-40.

Restriction MIDPOINTS= must be specified for the RANGE option to have any effect. Not supported by ActiveX.

RELZERO
creates area heights that are relative to a zero value. By default, GMAP creates heights that are relative to the minimum value, which might or might not be zero. With the RELZERO option, zero value areas have no height.

Alias REL0, RELATIVETOZERO

Restrictions This option works only for variables that have no negative values.

Not supported by Java

STATFMT=*format-specification*
overrides the GMAP default format for percent of PERCENT8.2. Use this format when using calculated values. The STATFMT option is typically used when the STATISTIC=FREQUENCY option or the PERCENT option is used.

Alias SFMT=, SFORMAT=, STATFORMAT=

STATISTIC=FIRST | SUM | FREQUENCY | MEAN
specifies the statistic for GMAP to chart. For character variables, FREQUENCY is the only allowed value—any other value is changed to FREQUENCY and a warning is issued. The frequency of a variable does not include missing values unless the MISSING option is specified.

FIRST
GMAP matches the first observation from the DATA= data set and charts the response value from this observation only. This is the default. If more rows exist that are not processed, a warning is issued to the log.

SUM
All observations matching a given ID value are added together and the summed value is charted.

FREQUENCY
A count of all rows with nonmissing values is charted unless you specify the MISSING option.

MEAN
All observations matching a given ID value are added together and then divided by the number of nonmissing observations matched. This value is then charted unless you specify the MISSING option.

Alias STAT=

STRETCH
stretches map extents to cover all available space in the device. This might cause the map to be distorted. When this option is applied to the PROC GMAP statement, it applies to all statements. If applied to a single statement, it applies only to that statement.

Alias STRETCHTOFIT, STR2FIT

Restriction Not supported by Java and ActiveX

UNIFORM
causes the same legend and coloring to be used for all maps produced by the procedure instead of being calculated within each BY group for each map. The UNIFORM option prescans the data to generate a categorization across all the data, regardless of BY grouping, and applies that categorization to all maps in the BY group. This results in all maps uniformly displaying legends and coloring such that a single value always has the same color in multiple maps.

When specified in a PROC GMAP statement, the UNIFORM option applies to all AREA, BLOCK, CHORO, and PRISM statements included within the GMAP run-group.

When omitted from the PROC GMAP statement, and specified on an individual AREA, BLOCK, CHORO, or PRISM statement, the UNIFORM option applies only to the maps produced by that statement.

Restriction Not supported by Java

URL=*character-variable*
specifies a character variable whose values are URLs. The variable values are URLs for web pages to display when the user clicks (drills down) on elements in the graph.

Restriction This option affects graphics output that is created through the ODS HTML destination only.

Interaction If you specify both the HTML= option and the URL= option, then the URL= option is ignored.

See "Overview of Enhancing Web Presentations" in *SAS/GRAPH: Reference*

"Example: GIF Output with Drill-Down Links" in *SAS/GRAPH: Reference*

WOUTLINE=*area-outline-width*
specifies the width, in pixels, of all map area outlines.

Default 1

XLIGHT=*x*
YLIGHT=*y*
specify the coordinates of the imaginary light source in the map coordinate system. The position of the light source affects how 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.

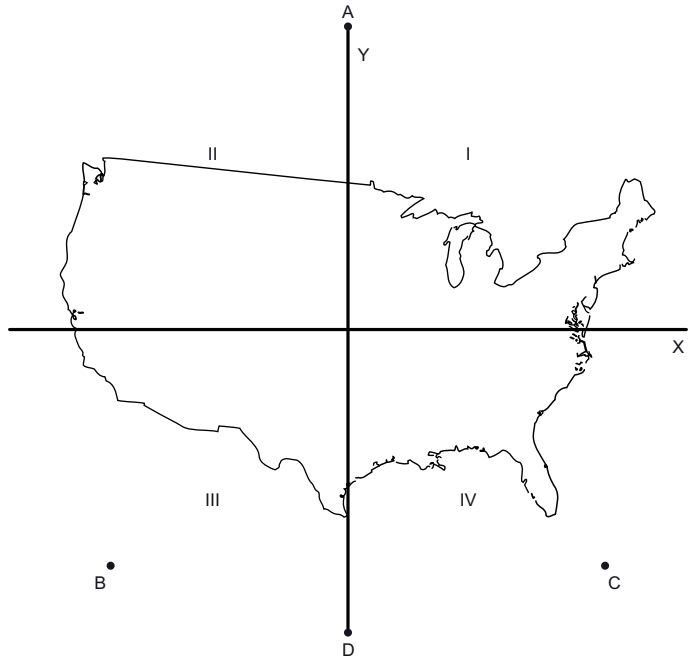
Table 7.5 on page 256 shows how the point that you specify is positioned.

Table 7.5 *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 7.6 on page 257 illustrates the light source positions. Assume that your viewing position, selected by the XVIEW=, YVIEW=, and ZVIEW= options, is point D.

Figure 7.6 *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.

Restriction Not supported by Java and ActiveX

See "XVIEW=*x* YVIEW=*y* ZVIEW=*z*" on page 257

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

specify the dimensions of the map that you are drawing. By default, the map uses the entire procedure output area.

Valid *units* are CELLS (character cells), CM (centimeters), IN (inches), or PCT (percentage of the graphics output area). The default unit is 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. If you specify one value and not the other, the dimension is adjusted to maintain the correct aspect ratio.

Restriction Not supported by Java and ActiveX

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).

The viewing position cannot coincide with the viewing reference point at coordinates (0.5, 0.5, 0).

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. One, two, or all three view coordinates can be specified; any that are not specified are assigned the default values.

Figure 7.5 on page 234 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 such a small ZVIEW= value that this condition cannot be satisfied, then PROC GMAP increases the ZVIEW= value automatically so that the angle is 45 degrees or less. Although you can use the XVIEW= and YVIEW= options with DEVICE=JAVA, ZVIEW= cannot be used with DEVICE=JAVA.

Alias	XV=, YV=, ZV=
Restriction	Partially supported by Java

Details

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 performs the following operations:

- determines the midpoints ranges or 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.

Note: PRISM maps do not work well with polygons within polygons (holes). It is recommended that a CHORO or BLOCK map be created for these maps instead.

SURFACE Statement

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

Restriction:	Not supported by Java and ActiveX
Requirement:	At least one response variable is required and must be numeric. The ID statement must be used in conjunction with the SURFACE statement.
Global statement:	TITLE, FOOTNOTE, NOTE

Syntax

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

Summary of Optional Arguments

Appearance options

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

CBODY=*surface-map-color*
: specifies the color that is used to draw the surface map.

CONSTANT=*n*
: specifies a denominator to use in the distance decay function.

NLINES=*number-of-lines*
: specifies the number of lines used to draw the surface map.

ROTATE=*degrees*
: specifies the degrees of the angle at which to rotate the map about the Z axis in the map coordinate system.

TILT=*degrees*
: specifies the degrees of the angle at which to tilt the map about the X axis in the map coordinate system.

XSIZE=*map-width <units>*

YSIZE=*map-height <units>*
: specify the physical dimensions of the map.

Description options

DESCRIPTION="*description*"
: specifies a description of the output.

NAME="*name*"
: specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

Mapping options

PERCENT
: causes GMAP to collect all response values (or their statistic) and chart each region as a percentage of the whole.

STATFMT=*format-specification*
: overrides the GMAP default format for percent of PERCENT8.2.

STATISTIC=FIRST | SUM | FREQUENCY | MEAN
: specifies the statistic for GMAP to chart.

Required Argument

response-variable(s)

specifies one or more variables in the response data set. The *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. Multiple response variables are separated 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 might not appear this way. The spikes in the front might appear to be higher than the spikes in the back, which represent greater values.

See "About Response Variables" on page 208.

Optional Arguments

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

ANNOTATE=*Annotate-data-set*
specifies a data set to annotate onto maps that are produced by the SURFACE statement. Annotate coordinate systems 1, 2, 7, and 8 are not valid with surface maps.

Alias ANNO=

See "Using Annotate Data Sets" in *SAS/GRAPH: Reference*

CBODY=*surface-map-color*
specifies the color that is used to draw the surface map. Regardless of the current ODS style, the default color is the first color in the current color list.

Alias CB=

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.

Alias CON=

DESCRIPTION="*description*"
specifies a description of the output. The maximum length for *description* is 256 characters. The description does not appear in the output. The descriptive text is shown in each of the following:

- the chart description for web output (depending on the device driver). See "Chart Descriptions for Web Presentations" in *SAS/GRAPH: Reference* for more information.
- the Table of Contents that is generated when you use the CONTENTS= option in an ODS HTML statement, assuming that the output is generated while the contents page is open.
- the description and the properties for the output in the Results window.
- the description and properties for the catalog entry in the Explorer.
- the **Description** field of the PROC GREPLAY window.

The *description* can include the #BYLINE, #BYVAL, and #BYVAR substitution options, which work as they do when used on TITLE, FOOTNOTE, and NOTE statements. Refer to "Substituting BY Line Values in a Text String" on page 73. The 256-character limit applies before the substitution takes place for these options. Thus, if in the SAS program the entry-description text exceeds 256 characters, it is truncated to 256 characters, and then the substitution is performed.

Alias DES=

Default SURFACE MAP OF *variable-name*

NAME="*name*"
specifies the name of the GRSEG catalog entry and the name of the graphics output file, if one is created.

The following applies to *name*:

- The name can be up to 256 characters in length.
- Special characters in the name are converted to underscores.
- **For the GRSEG entry name:**
  - The name is truncated to eight characters.
  - The first character is always represented in uppercase, and all other characters are represented in lowercase.
  - If the name begins with a number, an underscore is prepended to the name.
  - If the name duplicates an existing name, SAS/GRAPH appends a number or increments the last number used to create a unique graph name (for example, *name*1, *name*2, and so on). If necessary, the name is truncated so that the name and appended number do not exceed eight characters.
- **For the graphics output filename:**
  - The filename is based on the NAME= value except when you use an ODS LISTING destination, a DEVICE= option, and a file reference specifying an output filename. In this case, the file reference specification overrides the NAME= value. See "Controlling Graphics Output for ODS LISTING" in *SAS/GRAPH: Reference*.
  - All characters are represented in lowercase.
  - If a number is added to the GRSEG name, the same number is added to the output filename. See "About Filename Indexing" in *SAS/GRAPH: Reference*.
  - If the NAME= value is 8 characters or less, the filename is the GRSEG entry name.
  - If the NAME= value is greater than 8 characters, the NAME= value is used as the filename. When an index number is used in the GRSEG entry name, that index number is appended to the output filename. See "About Filename Indexing" in *SAS/GRAPH: Reference*.

  Note: If the name begins with a number, an underscore is prepended to the filename.
  - The maximum allowable filename length is device-specific. If the length of the name exceeds the maximum for the graphics device, an error results and no graphics output file is generated.

Default GMAP

NLINES=*number-of-lines*

specifies the number of lines used to draw the surface map. Values can range from 50 to 100; the higher the value, the more solid the map appears and the more resources used. By default, NLINES=50.

Alias N=

PERCENT

causes GMAP to collect all response values (or their statistic) and chart each region as a percentage of the whole. You can use the STATISTIC= option to change how the percentage is calculated—whether as a percentage of the SUM, FREQUENCY, or MEAN. If you do not use the STATISTIC= option, then STATISTIC=FIRST is assumed and the response variable of only the first observation of each region is

counted. If the response variable is a text field, then STATISTIC=FREQUENCY is used, even if you specify a different value for the STATISTIC= option.

Alias PERCENTAGE

See The "STATFMT=*format-specification*" on page 262, and the "STATISTIC=FIRST | SUM | FREQUENCY | MEAN" on page 262.

ROTATE=*degrees*
specifies the degrees of the angle at which to rotate the map about the Z axis in the map coordinate system. The *degrees* argument 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.

STATFMT=*format-specification*
overrides the GMAP default format for percent of PERCENT8.2. Use this format when using calculated values. The STATFMT option is typically used when the STATISTIC=FREQUENCY option or the PERCENT option is used.

Alias SFMT=, SFORMAT=, STATFORMAT=

STATISTIC=FIRST | SUM | FREQUENCY | MEAN
specifies the statistic for GMAP to chart. For character variables, FREQUENCY is the only allowed value—any other value is changed to FREQUENCY and a warning is issued. The frequency of a variable does not include missing values unless the MISSING option is specified.

FIRST
GMAP matches the first observation from the DATA= data set and charts the response value from this observation only. This is the default. If more rows exist that are not processed, a warning is issued to the log.

SUM
All observations matching a given ID value are added together and the summed value is charted.

FREQUENCY
A count of all rows with nonmissing values is charted unless you specify the MISSING option.

MEAN
All observations matching a given ID value are added together and then divided by the number of nonmissing observations matched. This value is then charted unless you specify the MISSING option.

Alias STAT=

TILT=*degrees*
specifies the degrees of the angle at which to tilt the map about the X axis in the map coordinate system. The value of *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, whereas TILT=0 corresponds to viewing the map from directly overhead. By default, TILT=70.

XSIZE=*map-width <units>*
YSIZE=*map-height <units>*
specify the physical dimensions of the map. By default, the map uses the entire procedure output area.

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

Details

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 enhance the map with an Annotate data set.

Using FIPS Codes and Province Codes

The map area identification variable in some SAS/GRAPH map data sets contain standardized numeric codes. The data sets for the United States contain a variable whose values are Federal Information Processing Standards (FIPS) codes. Traditional data sets for Canada contain standard province codes or census division codes. GfK data sets for Canada contain standard province codes and province names. You can specify a map data set with the GMAP procedure. However, ensure that the map area identification variable values in the map data set are the same as those used in the response data.

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 traditional map data sets supplied by SAS. Conversion might not be necessary if one of the GfK map data sets is used. This is because character variables such as **ID** that contain region values, might already match the identification variables in your response data set. Table 7.6 on page 263 lists the FIPS codes for the United States and Table 7.7 on page 264 lists the standard codes for Canadian provinces.

Note: Alternatively, you can convert the FIPS code or province codes in your traditional map data set to match the names in your response data.

Table 7.6 *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

FIPS code	State	FIPS code	State
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 7.7 *Canadian Province Codes*

Province code	Province
10	Newfoundland and Labrador

Province code	Province
11	Prince Edward Island
12	Nova Scotia
13	New Brunswick
24	Quebec
35	Ontario
46	Manitoba
47	Saskatchewan
48	Alberta
59	British Columbia
60	Yukon
61	Northwest Territories
62	Nunavut

Note: The *id-variables* are numeric in traditional but character in GfK Canadian map data sets.

The MAPS.CNTYNAME data set contains a cross-reference of names and FIPS codes for all counties in the United States. The MAPS.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 Functions and CALL Routines: Reference* for more information.

Table 7.8 *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 uppercase
STNAMEL	Converts state postal code to state name in mixed case
FIPNAME	Converts FIPS code to state name in uppercase
FIPNAMEL	Converts FIPS code to state name in mixed case
FIPSTATE	Converts FIPS code to state postal code

Using Formats for Traditional Map Data Set Variables

You can specify an output map area name or numeric value using one of the predefined formats for maps. The following prefixes are used in the names of the formats for maps:

CONT
: Continent

CNTRY
: Country

GLC
: Geographic Location Code, distributed by Government Services Administration, USA

ISO
: International Standard Organization

The formats for maps are located in the SASHELP.MAPFMTS catalog. See the MAPS.NAMES table to view all the continent and country names and corresponding GLC, ISO, and numeric representation for the continent values.

To use one of the formats for maps, you must specify the SASHELP.MAPFMTS catalog on the FMTSEARCH= option in a SAS OPTIONS statement:

```
options fmtsearch=(sashelp.mapfmts);
```

In addition to using the PUT statement (as shown in the examples in the following table), the formats can also be invoked using a FORMAT statement.

Note: If the input to a format is invalid, the format is “**” or “***” .

Table 7.9 *Formats for Maps*

FORMAT	DESCRIPTION	EXAMPLE	OUTPUT
contfmt	Use a continent's numeric value to output the continent's name	cont= 91 `put(cont,contfmt.);`	North America
glcna	Use the country's GLC numeric code to output the country's GLC alpha code	id=460 `put(id,glcna.);`	IR
glcnlu	Use the GLC numeric code to output the country's long name in uppercase	id=460 `put(id,glcnlu.);`	IRAN, ISLAMIC REPUBLIC OF
glcnsu	Use the GLC numeric code to output the country's short name in uppercase	id=460 `put(id,glcnsu.);`	IRAN

FORMAT	DESCRIPTION	EXAMPLE	OUTPUT
glcnsm	Use the GLC numeric code to output the country's name in mixed case	id=460 `put(id,glcnsm.);`	Iran
ison2a	Use the country's ISO numeric code to output the country's ISO alpha2 code	iso=364 `put(iso,ison2a.);`	IR
ison3a	Use the country's ISO numeric code to output the country's ISO alpha3 code	iso=364 `put(iso,ison3a.);`	IRN
isonlu	Use the country's ISO numeric code to output the country's long name in uppercase	iso=364 `put(iso,isonlu.);`	IRAN, ISLAMIC REPUBLIC OF
isonsu	Use the country's ISO numeric code to output the country's short name in uppercase	iso=364 `put(iso,isonsu.);`	IRAN
$cntrysl	Use a country's short name in uppercase to output the country's long name in uppercase	name='IRAN' `put(name,$cntrysl.);`	IRAN, ISLAMIC REPUBLIC OF
$glcalu	Use the GLC alpha code to output the country's long name in uppercase	country='IR' `put(country,$glcalu.);`	IRAN, ISLAMIC REPUBLIC OF
$glcan	Use the country's GLC alpha code to output the country's GLC numeric code	country='IR' `put(country,$glcan.);`	460
$glcsua	Use the country's short name in uppercase to output the GLC alpha code name	name='IRAN' `put(name,$glcsua.);`	IR
$glcsun	Use the country's short name in uppercase to output the country's GLC numeric code	name='IRAN' `put(name,$glcsun.);`	460
$glcsma	Use the country's short name in mixed-case to output the country's GLC alpha code	mixname='Iran' `put(mixname,$glcsma.);`	IR

FORMAT	DESCRIPTION	EXAMPLE	OUTPUT
$glcsmn	Use the country's short name in mixed-case to output the country's GLC numeric code	mixname='Iran' `put(mixname,$glcsmn.);`	460
$glcprov	Use a province or city name appended by || as a delimiter, followed by the country's GLC alpha code. This displays a province|| country code, the province or city code, and the country's GLC alpha numeric code	provname='TEHRAN||IR' `put(provname,$glcprov.);`	8250460 8250 — province or city code 460 — country GLC numeric code
$isosu2a	Use the country's short name in uppercase to display the country's ISO alpha2 code	name='IRAN' `put(name,$isosu2a.);`	IR
$isosu3a	Use the country's name in uppercase to display the country's ISO alpha3 code	name='IRAN' `put(name,$isosu3a.);`	IRN
$isosun	Use the country's short name in uppercase to display the country's ISO numeric code	name='IRAN' `put(name,$isosun.);`	364
$isoa2lu	Use the country's ISO alpha2 code to display the country's long name in uppercase	alpha2='IR' `put(alpha2,$isoa2su.);`	IRAN, ISLAMIC REPUBLIC OF
$isoa2n	Use the country's ISO alpha2 code to display the country's ISO numeric code	alpha2='IR' `put(alpha2,$isoa2n.);`	364
$isoa2su	Use the country's ISO alpha2 code to display the country's short name in uppercase	alpha2='IR' `put(alpha2,$isoa2lu.);`	IRAN
$isoa3lu	Use the country's ISO alpha3 code to display the country's long name in uppercase	alpha3='IRN' `put(alpha3,$isoa3lu.);`	IRAN, ISLAMIC REPUBLIC OF
$isoa3n	Use the country's ISO alpha3 code to display the country's ISO numeric code	alpha3='IRN' `put(alpha3,$isoa3n.);`	364

FORMAT	DESCRIPTION	EXAMPLE	OUTPUT
$isoa3su	Use the country's ISO alpha3 code to display the country's short name in uppercase	alpha3='IRN' `put(alpha3,$isoa3su.);`	IRAN
$isosm2a	Use the country's short name in mixed-case to display the country's ISO alpha2 code	mixname='Iran' `put(mixname,$isosm2a.);`	IR
$isosm3a	Use the country's short name in mixed-case to display the country's ISO alpha3 code	mixname='Iran' `put(mixname,$isosm3a.);`	IRN
$isosmn	Use the country's short name in mixed-case to display the country's ISO numeric code	mixname='Iran' `put(mixname,$isosmn.);`	364

Using SAS/GRAPH Map Data Sets

Specifying the Location of Map Data Sets

There are three library references (librefs) associated with the location of a SAS library; each library containing map data sets used by SAS/GRAPH. The three are as follows:

- MAPS
- MAPSGFK
- MAPSSAS

You can download the latest map data sets from the SAS Maps Online website at http://support.sas.com/rnd/datavisualization/mapsonline/index.html

MAPS references the library containing the latest set of SAS traditional maps. This library is defined in a configuration file with the value set to the same physical name (path) as MAPSSAS. In the following example, MAPS is the libref, and UK is a file in the SAS MAPS library.

```
MAPS.UK
```

The libref is assigned through the "MAPS= System Option" on page 81. It can also be set or changed with a LIBNAME statement, which enables each client to set their own definition for MAPS. This client setting would override the server setting defined at start-up time. Note that using the "APPEND= System Option" in *SAS System Options: Reference* or the "INSERT= System Option" in *SAS System Options: Reference*with the "MAPS= System Option" on page 81 allows an additional libref to be assigned to MAPS without changing its original definition. In this way existing programs can reference either the latest SAS or the MAPSGFK digital, vector map data sets.

MAPSGFK references the library containing the digital, vector map data sets at a location separate from the latest non-digital, vector SAS map data sets. This library is defined in a configuration file as a single physical path. This definition is set at installation time and should not be changed. The libref is assigned through the "MAPSGFK= System Option" on page 82.

MAPSSAS references the library containing the latest set of SAS traditional map files. This library is defined in a configuration file as a single physical name (path). It is the same path that MAPS references. This definition is set at installation time and should not be changed. The libref is assigned through the "MAPSSAS= System Option" on page 83.

Accessing Detailed Descriptions of Map Data Sets

You might need detailed information about the map data sets in order to determine their type, size, the variables that they contain, or, in the case of traditional map data sets, whether they are projected or unprojected. You can get this information by doing the following:

- using the CONTENTS or DATASETS procedure
- browsing the MAPS.METAMAPS data set in the MAPS library
- browsing the MAPSSAS.METAMAPS data set in the MAPSSAS library.
- browsing the library where your map data sets supplied by SAS reside

See "The METAMAPS Data Set" on page 204. Once the libref MAPS has been assigned, you can see a complete list of map data sets by viewing the MAPS.METAMAPS data set.

Run the following statements to list the map data sets in the SAS library that is assigned to the libref MAPS:

```
proc data sets lib=maps;
run;
```

Alternatively, you can substitute the libref MAPSGFK or MAPSSAS.

Run the following statements to provide detailed information about a map data set, including the number of observations, the variables in each data set, and a description of each variable:

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

To see the contents and descriptions of all of the map data sets associated with one of the three map libraries supplied by SAS, specify one of the following statements in the CONTENTS procedure:

- DATA=MAPS._ALL_
- DATA=MAPSSAS._ALL_
- DATA=MAPSGFK._ALL_

See the *Base SAS Procedures Guide* for more information about the CONTENTS and DATASETS procedures..

Customizing SAS/GRAPH Map Data Sets

Overview of Customizing 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 with the map data set to create a rectangular subset of a map data set. Do this by using the minimum and maximum longitude and latitude values in the map data set. For more information, see Chapter 8, "GPROJECT Procedure," on page 355.

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, and the longitude coordinates are measured in the same direction.

Map data sets supplied by SAS that contain coordinates expressed only as longitude and latitude, with variable names LONG and LAT, must be renamed X and Y. They should also be projected before you use them with the GMAP procedure. Map data sets from GfK GeoMarketing GmBH already contain projected coordinates expressed as longitude and latitude in degrees, with variable names LONG and LAT.

Subsetting Map Data Sets

Some of the SAS/GRAPH map data sets contain a large number of observations. Programs that use only a few states or provinces run faster if you exclude the unused portion of the map data set or use a reduced map data set. SAS 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 that you want.

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

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

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

Another approach is to use a DATA step to create a subset of the larger data set. This code illustrates a way to extract the observations for Quebec from the CANADA map 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. It is also useful when you need to make additional changes to the map data set.

See Chapter 10, "GREMOVE Procedure," on page 401 for an example of how to use GREMOVE to create a regional map from one of the map data sets that are supplied with SAS/GRAPH.

Reducing 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. It 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 9, "GREDUCE Procedure," on page 389.

Note: Many of the map data sets in the MAPS library are supplied with a DENSITY variable.

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. Therefore, 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).

You can set the DENSITY value by using the DENSITY= option in the PROC GMAP statement. For example, the following statement excludes all points with a density level of 2 or greater:

```
proc gmap map=maps.states 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 Traditional Map Data Sets

Traditional map data can be stored as unprojected or projected coordinates. GfK map data is stored as only projected. Unprojected map data contains spherical coordinates, that is, longitude and latitude values usually expressed in radians.[1]

Many of the map data sets in the MAPS library are projected. However, these map data sets contain only unprojected coordinates and should be projected before you use them.

- CANADA3
- CANADA4
- COUNTIES
- COUNTY
- STATES

If the projection supplied with the map data set does not meet your needs, then you can use the GPROJECT procedure (on unprojected map coordinates) to create a different projection. For more information about map data sets with unprojected coordinates, see "Map Data Sets Containing X, Y, LONG, and LAT" on page 195. You should select a projection method that least distorts the regions that you are mapping. (All projection methods inherently distort map regions.) See Chapter 8, "GPROJECT Procedure," on page 355 for more information.

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

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

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 when working with map data sets. In addition, displaying lakes might not be appropriate for some applications. In these cases, you might want to remove the lakes from the map data set before you proceed.

In traditional and GfK map data sets that contain coordinates for a lake that is located within a single internal division, the lake is identified with the numeric variable LAKE. The value of LAKE is 1 for points that correspond to lakes and 0 otherwise. STAT=FIRST processes only the first observation that matches the geographic region. The following statements illustrate how to delete the lakes from your traditional map data sets using WHERE processing:

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

The following statements illustrate how to delete the lakes from your GfK map data sets using WHERE processing:

```
proc gmap map=mapsgfk.us_all(where=(lake=0))
          data=mapsgfk.us_all;
   id id;
   choro id / levels=1 stat=first nolegend;
   title box=1 f=none h=4
         "U.S. with Lakes Removed";
run;
quit;
```

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

```
data nolake;
   set maps.chile(where=(lake=0));
run;
```

Creating SAS Map Data Sets

Creating Map Data Sets

Overview

In addition to using map data sets that are supplied with SAS/GRAPH software, you can also 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.

When creating map data sets, the following variables must be included:

- a numeric variable named X that contains the horizontal coordinates of the boundary points. The value of this variable is projected and represents longitude when your data is geographic. When your data is not geographic but defines spaces such as floor plans, this variable might not be projected and might not represent longitude. This variable is required by the GMAP procedure.
- a numeric variable named Y that contains the vertical coordinates of the boundary points. The value of this variable is projected and represents latitude when your data is geographic. When your data is not geographic but defines spaces such as floor plans, this variable might not be projected and might not represent latitude. This variable is required by the GMAP procedure.
- 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. An identification variable is required by the GMAP procedure.

The X and Y variable values in the map data set do not have to be in any specific units. 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.

The GMAP procedure uses the values of the X and Y variables to draw the map. To use the unprojected values to produce a custom map, follow the tasks in "Map Data Sets Containing X, Y, LONG, and LAT" on page 195.

Map data sets in which the X and Y variables contain unprojected longitude and latitude should be projected before you use them with PROC GMAP. See Chapter 8, "GPROJECT Procedure," on page 355 for details.

Here are optional variables to consider including in map data sets:

- a numeric variable named DENSITY that holds the density values returned from a GREDUCE procedure.
- a numeric variable named LONG that contains the horizontal coordinate of the boundary point in degrees or radians. The value of this variable is unprojected and represents longitude (east-west position).
- a numeric variable named LAT that contains the vertical coordinate of the boundary point in degrees or radians. The value of this variable is unprojected and represents latitude (north-south position).
- 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 is regarded as 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 connected. 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 the last point is joined to the first point to complete the polygon. All the segments for each ID value must be contiguous within the map data set. The observations must be specified in a clockwise direction. However, an enclosed polygon (a hole representing a lake, for example) must be specified in a counter-clockwise direction—opposite from the direction in which the containing polygon is drawn.
- a variable named RESOLUTION to map detail level based on output resolution. Refer to Table 7.3 on page 202 for a list of the output resolutions associated with a RESOLUTION variable value.

A unit area is defined by observations in the map data set that have the same identification (ID) variable value. A unit area might 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 within the same unit area.

- If the unit area is a single polygon, then all values of SEGMENT are the same (alternatively, you can omit the SEGMENT variable).
- If the unit area contains multiple polygons, such as islands, then 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 (holes), such as lakes, then the SEGMENT variable has one value but the interior polygon is defined by separate boundaries. To separate boundaries, a missing X and Y value must be inserted at the separation point. 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
;
```

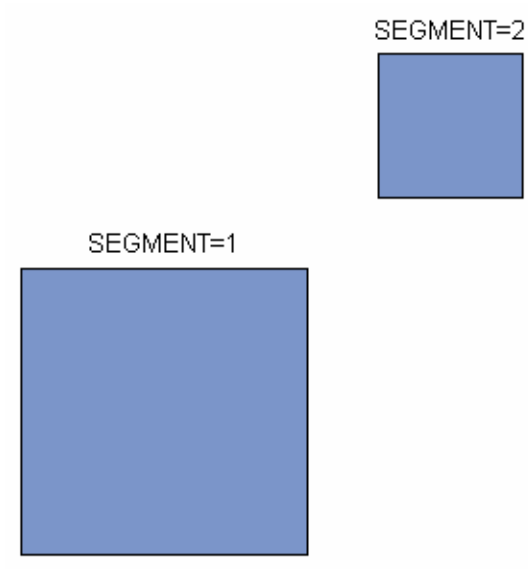
This data set does not have a SEGMENT variable.

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 7.7 on page 276:

```
data map;
   input id $ 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 7.7 *Single Unit Area with Two Segments (Polygons)*

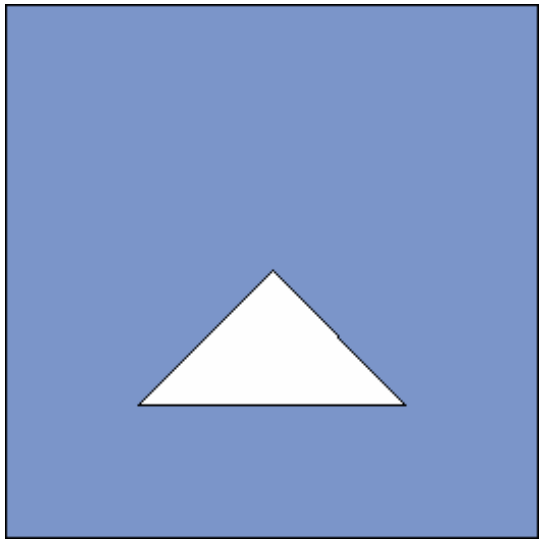


Creating a Unit Area That Contains Enclosed Polygons as Holes

Use separate boundaries to create an enclosed polygon (that is, a polygon that falls within the primary polygon for a single segment). The boundary for the hole is separated from the primary polygon boundary by inserting a missing value for X and Y. The last three observations are drawn counter-clockwise, in the opposite direction from the primary polygon. This enables an HTML image of the map to render correctly in all web browsers. For example, the data set that is created by this DATA step produces the map shown in Figure 7.8 on page 277:

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

Figure 7.8 *Single Unit Area with Hole*



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

Creating a Unit Area That Contains Another Area

Sometimes rather than a hole or lake, an enclosed polygon represents a separate map area. For example, in MAPS.AFRICA, the country of Lesotho is surrounded by the country of South Africa.

Note: As in some Traditional map data sets, MAPS.AFRICA specifies the coordinates to draw both country polygons in a clockwise direction. This might adversely affect the rendering of the HTML image of the map in some web browsers.

To create an enclosed map area:

1. Create an observation with missing values for X and Y for the surrounding area.
2. Define the boundary as part of the surrounding area by the using ID value for the surrounding area.
3. Define the boundary as part of the enclosed area by using the ID value for the enclosed area.

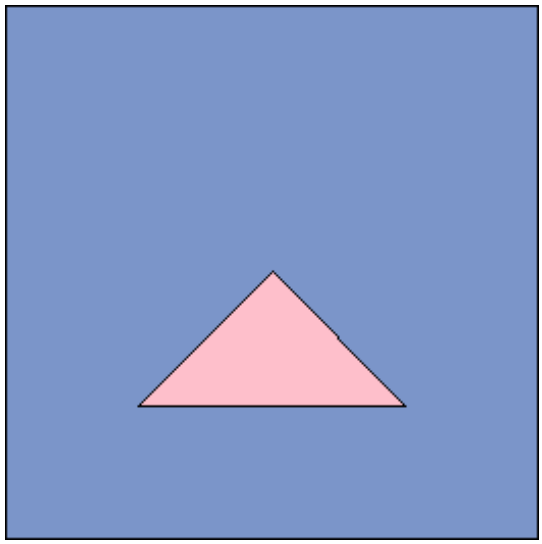
The triangle observations are drawn counter-clockwise, in the opposite direction from the hole in the primary (square) polygon. This enables an HTML image of the map to render correctly in all web browsers.

For example, this DATA step creates a data set that produces the map shown in Figure 7.9 on page 278:

```
data map;
   input id $ 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 3 1
triangle 1 2 2
triangle 1 1 1
;
```

Figure 7.9 *Unit Area within a Unit Area*



Using GfK Map Data Sets with Existing Code

Reworking Code That Uses Traditional Map Data Sets

Overview

This section presents several examples of existing code that uses traditional map data sets, with tips for repurposing the code to use GfK map data and variables. The existing code fragment is shown first, followed by the code rewritten with the recommended changes. The tips and details of the change are presented as **`/* comments */`** within each section of rewritten code.

Code to Prepare a Choropleth Map of Afghanistan

If you have existing code similar to the following:

```
proc gmap data=mydata map=maps.afghanis;
id id;
choro id / nolegend;
run;
```

Rewrite the code as follows to use a GfK map data set and ensure that the *id-variable*, in this case ID, is a matching character format between the response data set and the input data set:

```
/* Change the traditional map data set name (maps.afghanis) */
/* to the GfK map data set name (mapsgfk.afghanistan). */
/* The id-variable value in mydata must match the */
/* GfK id-variable character format. */
/* GfK id-variables ID and ID1 are character format. */
proc gmap data=mydata map=mapsgfk.afghanistan;
id id;
choro id / nolegend;
run;
```

Code to Prepare a Map of Europe for Annotation

If you have existing code similar to the following:

```
proc gmap data=mydata map=maps.europe;
id id;
choro id / levels=1
           nolegend
           anno=mypoints;
run;
```

Rewrite the code as follows to use a GfK map data set and a different *id-variable* because ID is now character format:

```
/* Change the traditional map data set name (maps.europe) */
/* to the GfK map data set name (mapsgfk.europe).          */
/* If the id-variable value in 'mydata' is numeric choose the */
/* GfK id-variable SEGMENT, which is also numeric format, or create */
/* your own variable. */
/* Note that GfK id-variables ID and ID1 are character format. */
/* The LEVELS=<number-of-midpoints> option will only work with numeric */
/* response variable values. */
proc gmap data=mydata map=mapsgfk.europe;
id segment;
choro segment / levels=1
          nolegend
          anno=mypoints;
run;
```

Code to Prepare a United States Map After Projecting Spherical Coordinates (Longitude and Latitude)

If you have existing code similar to the following, which projects the X and Y coordinate units in radians:

```
proc gproject data=maps.states out=mystates project=lambert;
id state;
run;

proc gmap data=mydata map=mystates;
id state;
choro state / nolegend;
run;
```

Rewrite the code as follows to use the GfK map data set and the projected X and Y coordinates (in degrees) provided in the input data set:

```
/* Change the traditional map data set name (maps.states) to the */
/* GfK map data set name mapsgfk.us_states. */
/* Remove Proc GPROJECT and use the provided projected coordinates in degrees. */
proc gmap data=mydata map=mapsgfk.us_states;
id state;
choro state / nolegend;
run;
```

Code to Project Annotate Data with a GfK Map Data Set

Use the following code to project only the Annotate data and not the map data already projected:

```
/* The mapsgfk.projparm data set contains input projection parameters. */

/* Project the points using the map's projection parameters stored that data set. */
/* The us_states entry in the projection parameters data set is used for  */
/* input parameters. */
 /* mapsgfk.us_states is the GfK map data set for the United States that */
/* contains unprojected longitude (X) and latitude (Y) variables in radians. */

proc gproject data=anno out=annoproj parmin=mapsgfk.projparm parmentry=us_states;
run;

proc gmap data=mydata map=mapsgfk.us_states anno=annoproj;
id state;
choro state / nolegend;
run;
```

Usage Tips with GfK Map Data Sets

Best Practices

When using the GfK map data set named **US_ALL**, the character identification variable **ID** uniquely refers to the state_county in the United States as well as other U.S. Territories. Specifying the variable **ID** instead of **STATE** ensures that the polygons are drawn correctly. For example:

```
proc gmap data=mydata map=mapsgfk.us_all;
id id;
choro id;
run;
```

If you want to use the **STATE** variable, use it in combination with the **COUNTY** identification variable:

```
proc gmap data=mydata map=mapsgfk.us_all;
id state county;
choro id;
run;
```

To more accurately create a map for the island nation of Maldives in the Indian Ocean, use the most current MAPSGFK.WORLD or the MAPSGFK.MALDIVES data sets. You might have existing code that uses the traditional MAPS.WORLD data set. In that

case, you can achieve the same level of map detail by using this data set version from the third maintenance release for SAS 9.1.

It is a requirement that each GfK map data set completes any enclosed polygons (holes) it contains in a counter-clockwise direction. This is the opposite direction from which the hole in the containing polygon is drawn. This enables an HTML map image to render correctly in all web browsers. It also enables any HTML hover-text to display correctly for an area containing a hole without interfering with that hole.

Note: Traditional map data sets vary in the direction in which they complete enclosed polygons (holes). For example, the MAPS.AFRICA data set completes enclosed polygons in a clockwise direction. Traditional map data sets were created before the setting of map data set requirements to enable viewing HTML in web browsers.

Examples: GMAP Procedure

Example 1: Using GfK GeoMarketing Map Data to Produce a Simple Block Map

Features: MAP= required argument referring to GfK map data

DATA= argument referring to response data

ID statement

RESOLUTION= option

BLOCK statement options
- BLOCKSIZE=
- RELZERO

Other features: System option FMTSEARCH=

SQL procedure

PATTERN statement

Data sets: MAPSGFK.ASIA (map data)

DEMOGRAPHICS (table of response data)

Format: ison2a

Restriction: The GfK GeoMarketing map data set used in this example is licensed to be used only with SAS/GRAPH.

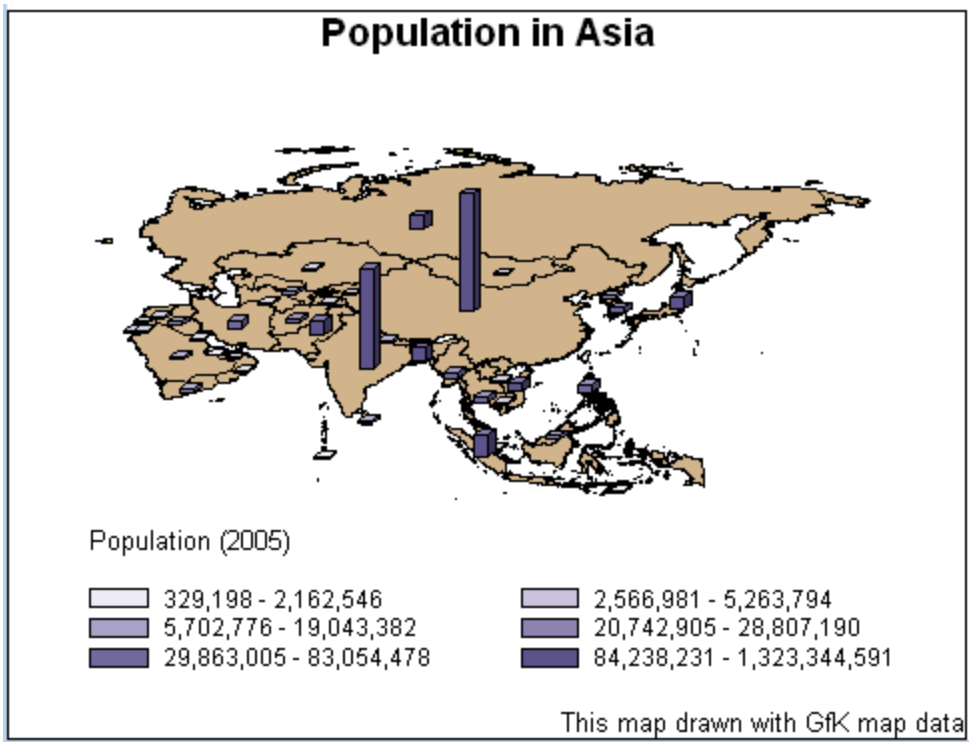
Sample library member: GMPGSIMP

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example uses GfK map data to produce a block map that shows the population of countries in Asia. Because 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. It then calculates the appropriate response levels. The elongated countries in the displayed map indicate that a different projection method was used with this GfK map than was used with the traditional Asia map data set.



Program

```
options fmtsearch=(sashelp.mapfmts);

proc sql;
create table demographics(rename=(iso=oiso newiso=iso id=oldid newid=ID)) as
select demo.*,
put(demo.iso,z3.) as newiso format=$3.,
put(demo.iso,ison2a.) as newid
from sashelp.demographics as demo
;
alter table demographics
modify ID char(15) label='Alpha2 Country Code';
quit;

goptions reset=all border;

title1 "Population in Asia";
legend1 label=(position=top) shape=bar(.3in,.1in);
footnote1 j=r "This map drawn with GfK map data";
pattern1 value=msolid color=tan;

proc gmap data=demographics(where=(cont=95))
           map=mapsgfk.asia resolution=4 all;
```

```
   id iso;
   block pop / blocksize=1 relzero legend=legend1;
run;
quit;
```

Program Description

Specify the format catalog to search that has the predefined ISO alpha2 code.

```
options fmtsearch=(sashelp.mapfmts);
```

Create a table named demographics using sashelp.demographics as the base and changing its variables to match GfK variable types and lengths. This table is used as the response data set. Note that the ISO variable was numeric in the original sashelp.demographics data but is the character variable OISO in the GfK map data set. The format 'ison2a' uses the country's ISO numeric code to display the country's ISO alpha2 code. Also note that the ID variable was a numeric geographic locator code (glc) in the original sashelp.demographics data. However, it is represented by the ISOALPHA2 variable in the GfK map data set.

```
proc sql;
create table demographics(rename=(iso=oiso newiso=iso id=oldid newid=ID)) as
select demo.*,
put(demo.iso,z3.) as newiso format=$3.,
put(demo.iso,ison2a.) as newid
from sashelp.demographics as demo
;
alter table demographics
modify ID char(15) label='Alpha2 Country Code';
quit;
```

Set the graphics environment.

```
goptions reset=all border;
```

Define the title, legend, footnote, and color for the map. The PATTERN statement specifies a solid tan color pattern for the Asian continent, differentiating it from the block colors.

```
title1 "Population in Asia";
legend1 label=(position=top) shape=bar(.3in,.1in);
footnote1 j=r "This map drawn with GfK map data";
pattern1 value=msolid color=tan;
```

Produce the block map. Specify the demographics table just created as the response data set. Specify a GfK map data set named mapsgfk.asia. Specifically use the Asia continent. The RESOLUTION= option specifies that all map observations with a RESOLUTION variable value of 4 or less be used. The 4 value corresponds to a resolution of 800 x 600 pixels. ALL specifies that the generated map should include all the map areas from the map data set, even if there is no observation for that map area. The ID statement specifies the variable that is in both the map data set and the response data set and defines map areas. Note that the ISO variable is character. The BLOCK statement specifies the population variable in the response data set that contains the response values for each of the map areas. The BLOCKSIZE= option specifies the width of the blocks. The RELZERO option specifies that the block values are relative to zero.

```
proc gmap data=demographics(where=(cont=95))
```

```
            map=mapsgfk.asia resolution=4 all;
   id iso;
   block pop / blocksize=1 relzero legend=legend1;
run;
quit;
```

Example 2: Using Traditional Map Data to Produce a Simple Block Map

Features: MAP= required argument referring to traditional map data

DATA= required argument referring to response data

ID statement

BLOCK statement options
- BLOCKSIZE=
- RELZERO

Other features: PATTERN statement

Data sets: MAPS.ASIA (map data)

SASHELP.DEMOGRAPHICS (response data)

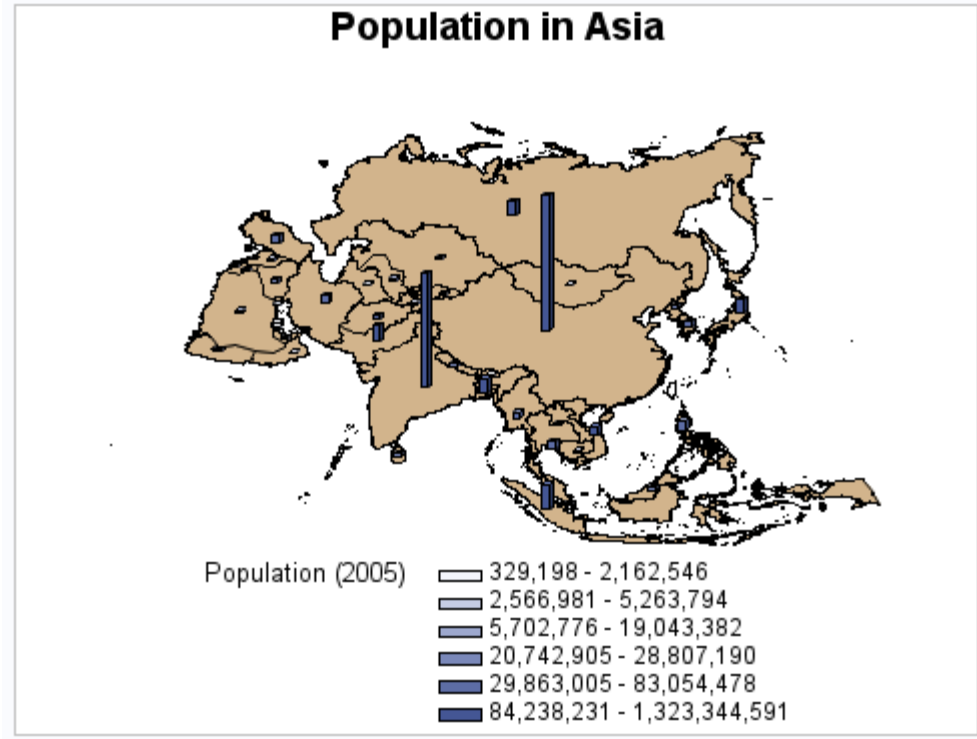
Sample library member: GMPSIMPL

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example produces a block map that shows population of countries in Asia. Because 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. It then calculates the appropriate response levels.



Program

```
goptions reset=all border;

title1 "Population in Asia";
pattern1 value=msolid color=tan;

proc gmap data=sashelp.demographics(where=(cont=95))
          map=maps.asia all;
   id id;
   block pop / blocksize=1 relzero;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the title and the color for the map. The PATTERN statement specifies a solid tan color pattern for the Asian continent, differentiating it from the block colors.

```
title1 "Population in Asia";
pattern1 value=msolid color=tan;
```

Produce the block map. The ALL argument specifies that the output should include all of the map areas from the map data set. This happens even if the response data set SASHELP.DEMOGRAPHICS does not include an observation for the map area. 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. The BLOCKSIZE=

option specifies the width of the blocks. The RELZERO option specifies that the block values are relative to zero.

```
proc gmap data=sashelp.demographics(where=(cont=95))
          map=maps.asia all;
   id id;
   block pop / blocksize=1 relzero;
run;
quit;
```

Example 3: Using GfK GeoMarketing Map Data to Specify Response Levels in a Block Map

Features: MAP= required argument referring to GfK map data

DATA= argument referring to response data

ID statement

BLOCK statement options
- LEVELS=
- SHAPE=
- RELZERO
- CEMPTY=

Other features: System option FMTSEARCH=

SQL procedure

PATTERN statement

Data sets: MAPSGFK.SAMERICA (map data)

DEMOGRAPHICS (table of response data)

Format: ison2a

Restriction: The GfK GeoMarketing map data set used in this example is licensed to be used only with SAS/GRAPH.

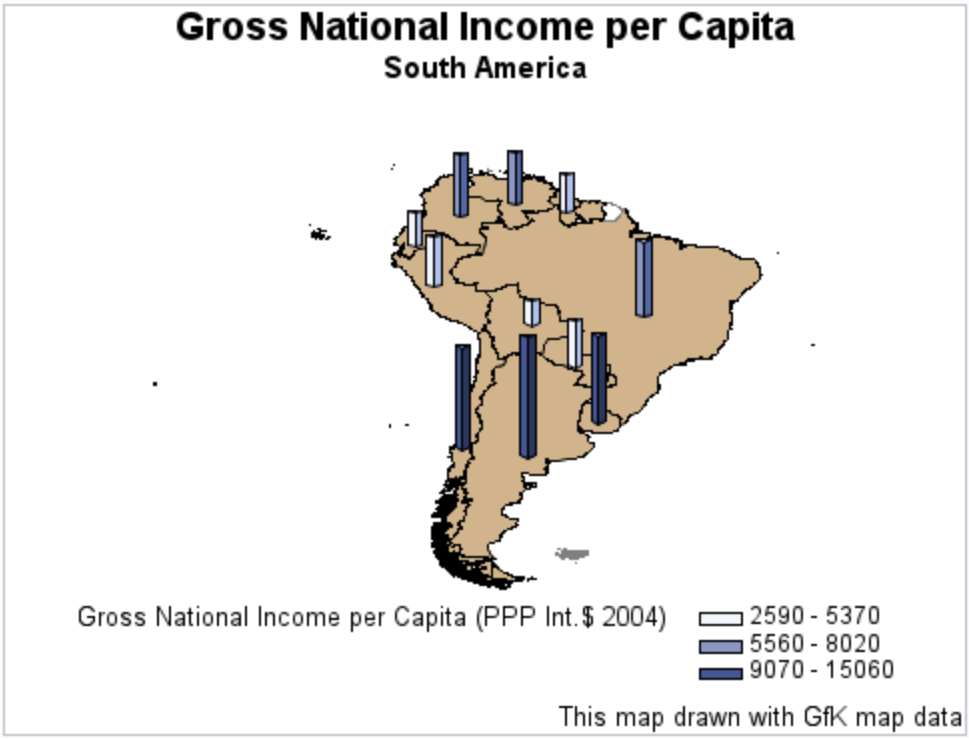
Sample library member: GMPGLEVL

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example uses the LEVELS= option to specify the number of response levels for the blocks. The LEVELS= option tells GMAP how many response levels and the GMAP procedure calculates the quantiles.



Program

```
options fmtsearch=(sashelp.mapfmts);

proc sql;
create table demographics(rename=(iso=oiso newiso=iso id=oldid newid=ID)) as
select demo.*,
put(demo.iso,z3.) as newiso format=$3.,
put(demo.iso,ison2a.) as newid
from sashelp.demographics as demo
;
alter table demographics
modify ID char(15) label='Alpha2 Country Code';
quit;

goptions reset=all border;

title1 "Gross National Income per Capita";
title2 "South America";
footnote1 j=r "This map drawn with GfK map data";
pattern1 value=msolid color=tan;

proc gmap data=demographics(where=(cont=92))
map=mapsgfk.samerica all;
id iso;
block gni / levels=3 shape=prism
            relzero cempty=gray;
run;
quit;
```

Program Description

Specify the format catalog to search that has the predefined ISO alpha2 code.

```
options fmtsearch=(sashelp.mapfmts);
```

Create a table named demographics using sashelp.demographics as the base and changing its variables to match GfK variable types and lengths. This table is used as the response data set. Note that the ISO variable was numeric in the original sashelp.demographics data but is the character variable OISO in the GfK map data set. The format 'ison2a' uses the country's ISO numeric code to display the country's ISO alpha2 code. Also note that the ID variable was a numeric geographic locator code (glc) in the original sashelp.demographics data. However, it is represented by the ISOALPHA2 variable in the GfK map data set.

```
proc sql;
create table demographics(rename=(iso=oiso newiso=iso id=oldid newid=ID)) as
select demo.*,
put(demo.iso,z3.) as newiso format=$3.,
put(demo.iso,ison2a.) as newid
from sashelp.demographics as demo
;
alter table demographics
modify ID char(15) label='Alpha2 Country Code';
quit;
```

Set the graphics environment.

```
goptions reset=all border;
```

Define the titles, footnote, and color for the map. The PATTERN statement specifies a solid tan color pattern for the South American continent, differentiating it from the block colors.

```
title1 "Gross National Income per Capita";
title2 "South America";
footnote1 j=r "This map drawn with GfK map data";
pattern1 value=msolid color=tan;
```

Produce the block map. The ALL argument specifies that the output should include all of the map areas from the map data set. This happens even if the response data set DEMOGRAPHICS does not include an observation for the map area. The LEVELS= option specifies the number of response levels for the graph. The SHAPE= option draws the blocks as prisms. The RELZERO option specifies that the block values are relative to zero. The CEMPTY= option specifies the outline color for map areas that have missing data. Note that the block map does not include the Caribbean islands. GfK offers the map data for those islands in a separate data set.

```
proc gmap data=demographics(where=(cont=92))
map=mapsgfk.samerica all;
id iso;
block gni / levels=3 shape=prism
            relzero cempty=gray;
run;
quit;
```

Example 4: Using Traditional Map Data to Specify Response Levels in a Block Map

Features: MAP= required argument referring to traditional map data

DATA= required argument referring to response data

ID statement

BLOCK statement options
- LEVELS=
- SHAPE=
- RELZERO
- CEMPTY=

Other features: PATTERN statement

Data sets: MAPS.SAMERICA (map data)

SASHELP.DEMOGRAPHICS (response data)

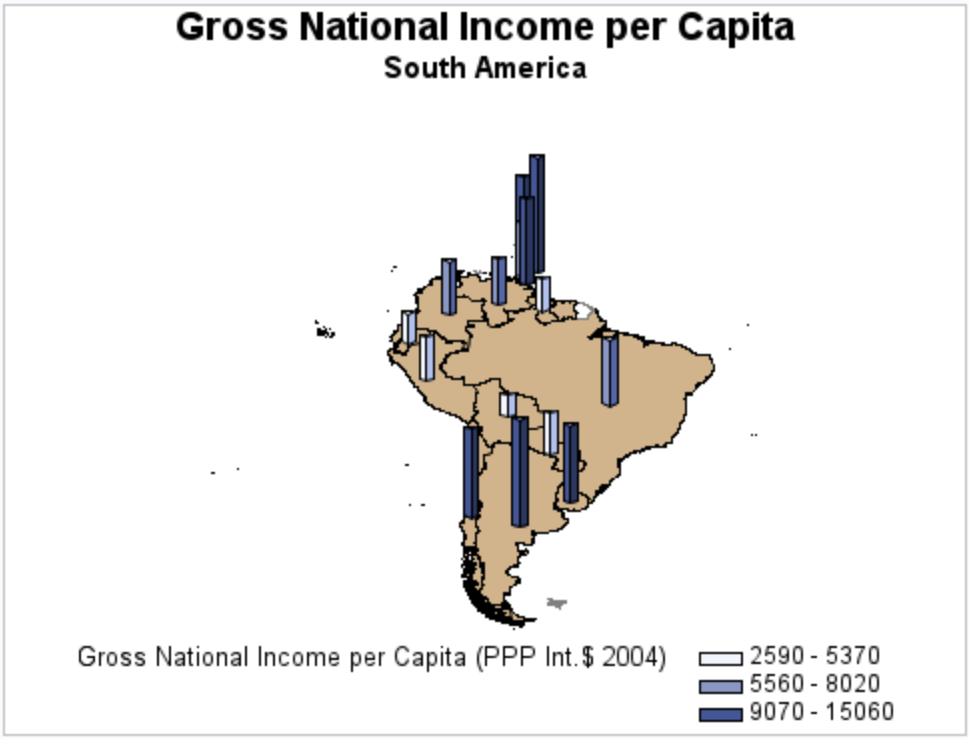
Sample library member: GMPLEVEL

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example uses the LEVELS= option to specify the number of response levels for the blocks. The LEVELS= option tells GMAP how many response levels and the GMAP procedure calculates the quantiles.



Program

```
goptions reset=all border;
```

```
title1 "Gross National Income per Capita";
title2 "South America";
pattern1 value=msolid color=tan;

proc gmap data=sashelp.demographics(where=(cont=92))
          map=maps.samerica all;
   id id;
   block gni / levels=3 shape=prism
               relzero cempty=gray;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the titles, footnote, and color for the map. The PATTERN statement specifies a solid tan color pattern for the South American continent, differentiating it from the block colors.

```
title1 "Gross National Income per Capita";
title2 "South America";
pattern1 value=msolid color=tan;
```

Produce the block map, which includes the Caribbean islands. The ALL argument specifies that the output should include all of the map areas from the map data set. This happens even if the response data set DEMOGRAPHICS does not include an observation for the map area. The LEVELS= option specifies the number of response levels for the graph. The SHAPE= option draws the blocks as prisms. The RELZERO option specifies that the block values are relative to zero. The CEMPTY= option specifies the outline color for map areas that have missing data.

```
proc gmap data=sashelp.demographics(where=(cont=92))
          map=maps.samerica all;
   id id;
   block gni / levels=3 shape=prism
               relzero cempty=gray;
run;
quit;
```

Example 5: Using GfK GeoMarketing Map Data to Assign a Format to the Response Variable

Features: MAP= required argument referring to GfK map data

DATA= argument referring to response data

FORMAT statement

ID statement

AREA statement option MIDPOINTS=

BLOCK statement options

- LEGEND=
- RELZERO
- LEVELS=

Other features: System option FMTSEARCH=
SQL procedure
FORMAT procedure
LEGEND statement
PATTERN statement

Data sets: MAPSGFK.ASIA (map data)
DEMOGRAPHICS (table of response data)

Format: ison2a

Restriction: The GfK GeoMarketing map data set used in this example is licensed to be used only with SAS/GRAPH.

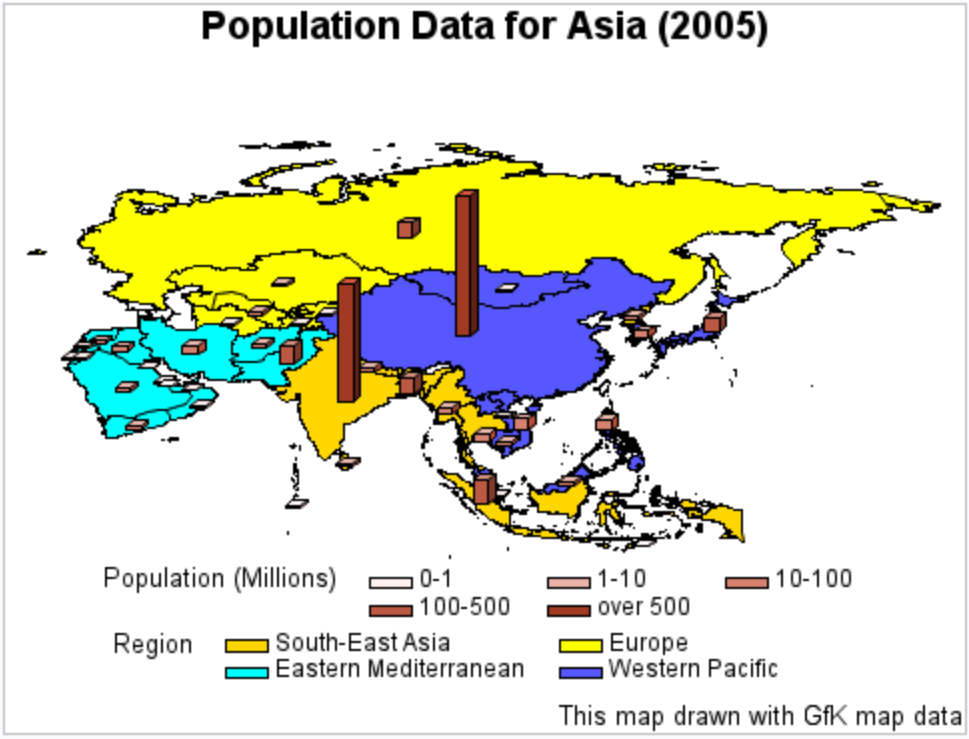
Sample library member: GMPGFRMT

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example creates formats for the response variables. The format for the POP variable defines and labels ranges of values. These ranges appear in the legend and make the map easier to understand. The example also uses the AREA statement to patterns the map areas by region. The elongated countries in the output indicate that a different projection method was used with this GfK map than was used with the traditional Asia map data set.



Program

```
options fmtsearch=(sashelp.mapfmts);

proc sql;
create table demographics(rename=(iso=oiso newiso=iso id=oldid newid=ID)) as
select demo.*,
put(demo.iso,z3.) as newiso format=$3.,
put(demo.iso,ison2a.) as newid
from sashelp.demographics as demo
;
alter table demographics
modify ID char(15) label='Alpha2 Country Code';
quit;

goptions reset=all border;

proc format;
  value popfmt low-1000000="0-1"
               1000001-10000000="1-10"
               10000001-100000000="10-100"
               100000001-500000000="100-500"
               500000001-high="over 500";
run;

proc format;
  value $ regionfmt "SEAR" = "South-East Asia"
                    "EUR" = "Europe"
                    "EMR" = "Eastern Mediterranean"
                    "WPR" = "Western Pacific";
run;

title1 "Population Data for Asia (2005)";
footnote j=r "This map drawn with GfK map data";
pattern1 color=gold value=msolid;
pattern2 color=yellow value=msolid;
pattern3 color=cyan value=msolid;
pattern4 color=light_blue value=msolid;

legend1 label=("Population (Millions)");

proc gmap data=demographics(where=(cont=95))
map=mapsgfk.asia all;
format pop popfmt.;
format region $regionfmt.;
id id;
area region / midpoints="SEAR" "EUR" "EMR" "WPR";
block pop / legend=legend1
            relzero
            levels=all;
run;
quit;
```

Program Description

Specify the format catalog to search that has the predefined ISO alpha2 code.

```
options fmtsearch=(sashelp.mapfmts);
```

Create a table named demographics using sashelp.demographics as the base and changing its variables to match GfK variable types and lengths. This table is used as the response data set. Note that the ISO variable was numeric in the original sashelp.demographics data but is the character variable OISO in the GfK map data set. The format 'ison2a' uses the country's ISO numeric code to display the country's ISO alpha2 code. Also note that the ID variable was a numeric geographic locator code (glc) in the original sashelp.demographics data. However, it is represented by the ISOALPHA2 variable in the GfK map data set.

```
proc sql;
create table demographics(rename=(iso=oiso newiso=iso id=oldid newid=ID)) as
select demo.*,
put(demo.iso,z3.) as newiso format=$3.,
put(demo.iso,ison2a.) as newid
from sashelp.demographics as demo
;
alter table demographics
modify ID char(15) label='Alpha2 Country Code';
quit;
```

Set the graphics environment.

```
goptions reset=all border;
```

Create a format for the POP variable. POPFMT. defines the ranges of values for POP and labels the values.

```
proc format;
  value popfmt low-1000000="0-1"
               1000001-10000000="1-10"
               10000001-100000000="10-100"
               100000001-500000000="100-500"
               500000001-high="over 500";
run;
```

Create a format for the REGION variable. REGIONFMT. labels the values for REGION.

```
proc format;
  value $ regionfmt "SEAR" = "South-East Asia"
                    "EUR" = "Europe"
                    "EMR" = "Eastern Mediterranean"
                    "WPR" = "Western Pacific";
run;
```

Define the title, footnote, and colors for the map. The PATTERN statement specifies a unique solid color pattern for each country of the Asian continent, differentiating it from the block colors.

```
title1 "Population Data for Asia (2005)";
footnote j=r "This map drawn with GfK map data";
pattern1 color=gold value=msolid;
pattern2 color=yellow value=msolid;
pattern3 color=cyan value=msolid;
pattern4 color=light_blue value=msolid;
```

Assign the legend label.

```
legend1 label=("Population (Millions)");
```

Produce the block map. The ALL argument specifies that the map displayed should include all of the map areas from the map data set. This happens even if the response data set DEMOGRAPHICS does not include an observation for the map area. The map displayed shows one such area. The FORMAT statements assign POPFMT. to the POP variable and $REGIONFMT. to the REGION variable. The ID statement assigns the variable ID—which represents the ISOALPHA2 variable after the variable rename when the demographics table was set up. The AREA statement assigns patterns to the map areas according to the values of the REGION variable. The RELZERO option specifies that the blocks values are relative to zero. The LEVELS= option is used to graph all unique ID response variable values.

```
proc gmap data=demographics(where=(cont=95))
map=mapsgfk.asia all;
format pop popfmt.;
format region $regionfmt.;
id id;
area region / midpoints="SEAR" "EUR" "EMR" "WPR";
block pop / legend=legend1
            relzero
            levels=all;
run;
quit;
```

Example 6: Using Traditional Map Data to Assign a Format to the Response Variable

Features: MAP= required argument referring to traditional map data
DATA= required argument referring to response data
FORMAT statement
ID statement
AREA statement option MIDPOINTS=
BLOCK statement options
- LEGEND=
- RELZERO

Other features: FORMAT procedure
LEGEND statement
PATTERN statement

Data sets: MAPS.ASIA (map data)
SASHELP.DEMOGRAPHICS (response data)

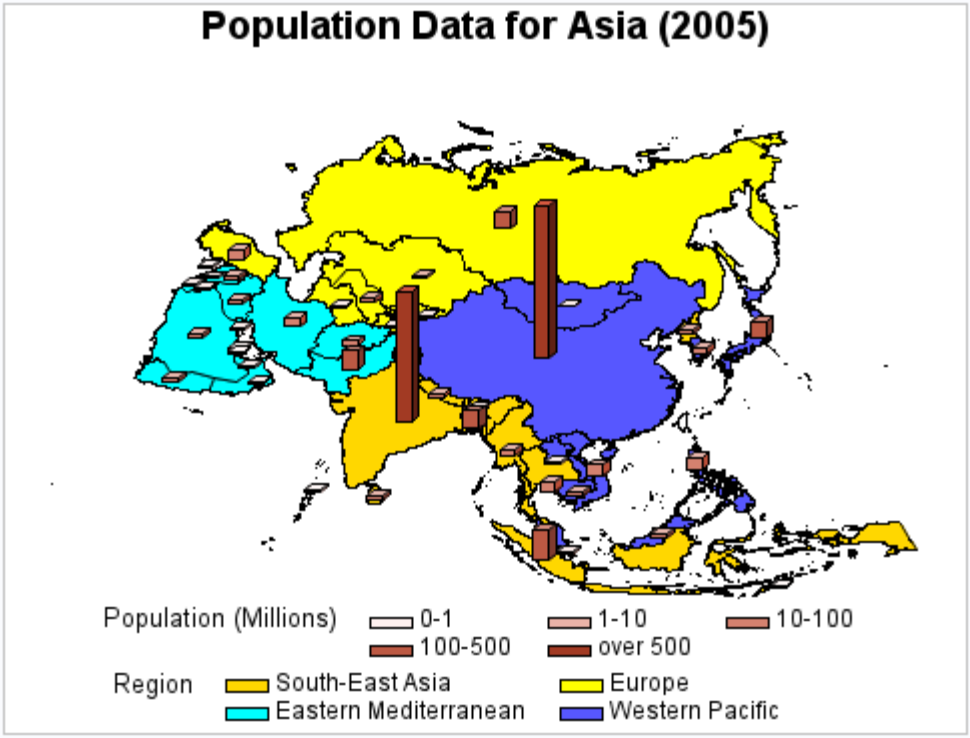
Sample library member: GMPFORMT

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example creates formats for the response variables. The format for the POP variable defines and labels ranges of values. These ranges appear in the legend and make the map easier to understand. The example also uses the AREA statement to patterns the map areas by region.



Program

```
goptions reset=all border;

proc format;
  value popfmt low-1000000="0-1"
               1000001-10000000="1-10"
               10000001-100000000="10-100"
               100000001-500000000="100-500"
               500000001-high="over 500";
run;

proc format;
  value $ regionfmt "SEAR" = "South-East Asia"
                    "EUR" = "Europe"
                    "EMR" = "Eastern Mediterranean"
                    "WPR" = "Western Pacific";
run;

title1 "Population Data for Asia (2005)";
pattern1 color=gold value=msolid;
pattern2 color=yellow value=msolid;
pattern3 color=cyan value=msolid;
pattern4 color=light_blue value=msolid;

legend1 label=("Population (Millions)");

proc gmap data=sashelp.demographics(where=(cont=95))
```

```
            map=maps.asia all;
    format pop popfmt.;
    format region $regionfmt.;
    id id;
    area region / midpoints="SEAR" "EUR" "EMR" "WPR";
    block pop / legend=legend1
                relzero
                levels=all;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create a format for POP. POPFMT. defines the ranges of values for POP and labels the values.

```
proc format;
  value popfmt low-1000000="0-1"
               1000001-10000000="1-10"
               10000001-100000000="10-100"
               100000001-500000000="100-500"
               500000001-high="over 500";
run;
```

Create a format for REGION. REGIONFMT. labels the values for REGION.

```
proc format;
  value $ regionfmt "SEAR" = "South-East Asia"
                    "EUR" = "Europe"
                    "EMR" = "Eastern Mediterranean"
                    "WPR" = "Western Pacific";
run;
```

Define the title and colors for the map. The PATTERN statement specifies a unique solid color pattern for each country of the Asian continent, differentiating it from the block colors.

```
title1 "Population Data for Asia (2005)";
pattern1 color=gold value=msolid;
pattern2 color=yellow value=msolid;
pattern3 color=cyan value=msolid;
pattern4 color=light_blue value=msolid;
```

Assign the legend label.

```
legend1 label=("Population (Millions)");
```

Produce the block maps. The ALL argument specifies that the map displayed should include all of the map areas from the map data set. This happens even if the response data set DEMOGRAPHICS does not include an observation for the map area. The FORMAT statements assign POPFMT. to the POP variable and $REGIONFMT. to the REGION variable. The AREA statement assigns patterns to the map areas according to the values of the REGION variable. The RELZERO option specifies that the blocks

values are relative to zero. The LEVELS= option is used to graph all unique ID response variable values.

```
proc gmap data=sashelp.demographics(where=(cont=95))
          map=maps.asia all;
   format pop popfmt.;
   format region $regionfmt.;
   id id;
   area region / midpoints="SEAR" "EUR" "EMR" "WPR";
   block pop / legend=legend1
               relzero
               levels=all;
run;
quit;
```

Example 7: Using GfK GeoMarketing Map Data When Specifying the Statistic for the Response Variable

Features: MAP= required argument referring to GfK map data

DATA= required argument referring to response data

ID statement

BLOCK statement options
- STATISTIC=
- LEVELS=
- RELZERO

Other features: PATTERN statement

Data sets: MAPSGFK.US (map data)

SASHELP.ZIPCODE (response data)

Restriction: The GfK GeoMarketing map data set used in this example is licensed to be used only with SAS/GRAPH.

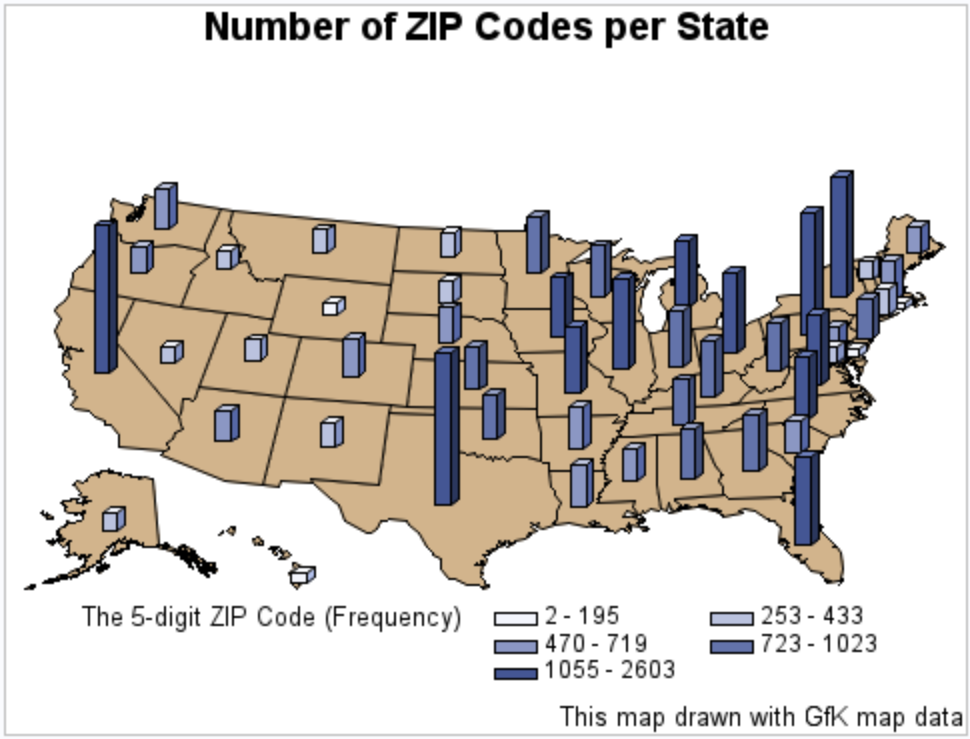
Sample library member: GMPGSTAT

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example specifies the statistic for the response variable that is displayed by the block map. The STATISTIC= option specifies that the statistic is frequency rather than the default statistic (sum).



Program

```
goptions reset=all border;

title1 "Number of ZIP Codes per State";
footnote j=r "This map drawn with GfK map data";
pattern1 value=msolid color=tan;

proc gmap map=mapsgfk.us data=sashelp.zipcode all;
id state;
block zip / statistic=frequency
            levels=5
            relzero;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the title, footnote, and color for the map.

```
title1 "Number of ZIP Codes per State";
footnote j=r "This map drawn with GfK map data";
pattern1 value=msolid color=tan;
```

Produce the block maps. The ALL argument specifies that the map displayed should include all of the map areas from the map data set. This happens even if the response data set DEMOGRAPHICS does not include an observation for the map area. The STATISTIC= option specifies the statistic for the response variable. The LEVELS= option specifies the number of response levels. The RELZERO option specifies that the

blocks values are relative to zero. Note that the MAPSGFK.US map data set does not include Puerto Rico or the District of Columbia. Block maps created with the traditional MAPS.US map data set include these map areas.

```
proc gmap map=mapsgfk.us data=sashelp.zipcode all;
id state;
block zip / statistic=frequency
            levels=5
            relzero;
run;
quit;
```

Example 8: Using Traditional Map Data When Specifying the Statistic for the Response Variable

Features: MAP= required argument referring to traditional map data

DATA= required argument referring to response data

ID statement

BLOCK statement options
- STATISTIC=
- LEVELS=
- RELZERO

Other features: PATTERN statement

Data sets: MAPS.US (map data)

SASHELP.ZIPCODE (response data)

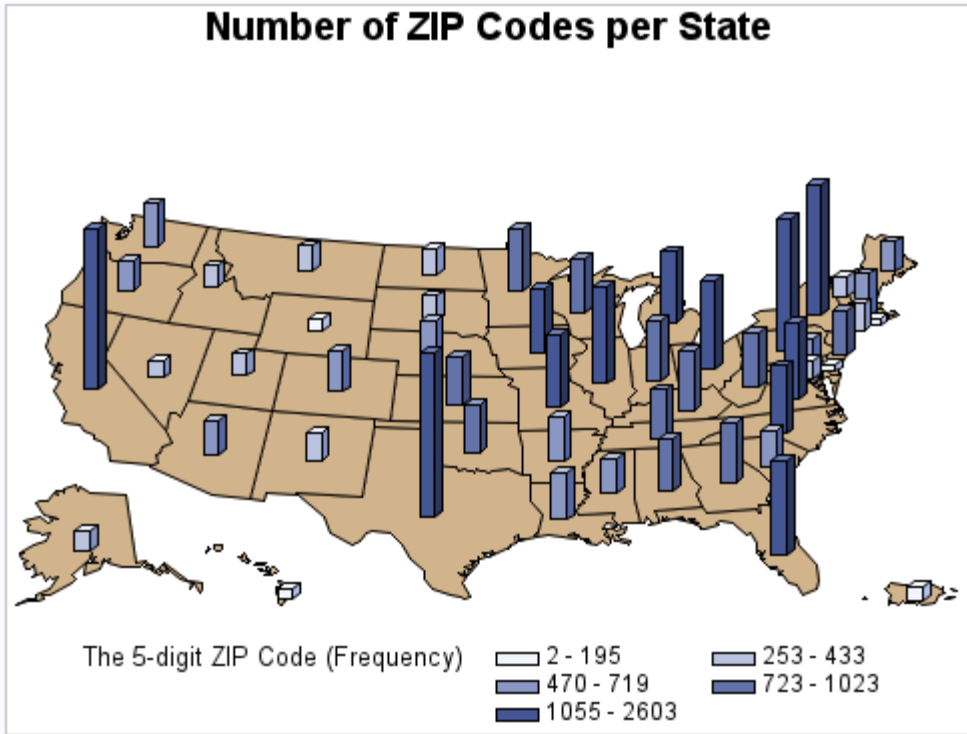
Sample library member: GMPSTAT

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example specifies the statistic for the response variable that is displayed by the block map. The STATISTIC= option specifies that the statistic is frequency rather than the default statistic (sum).



Program

```
goptions reset=all border;

title1 "Number of ZIP Codes per State";
pattern1 value=msolid color=tan;

proc gmap map=maps.us data=sashelp.zipcode all;
   id state;
   block zip / statistic=frequency
               levels=5 relzero;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the title and the color for the map.

```
title1 "Number of ZIP Codes per State";
pattern1 value=msolid color=tan;
```

Produce the block maps. These include Puerto Rico and the District of Columbia. The ALL argument specifies that the map displayed should include all of the map areas from the map data set. This happens even if the response data set DEMOGRAPHICS does not include an observation for the map area. The STATISTIC= option specifies the statistic for the response variable. The LEVELS= option specifies the number of response levels. The RELZERO option specifies that the blocks values are relative to zero.

```
proc gmap map=maps.us data=sashelp.zipcode all;
   id state;
   block zip / statistic=frequency
               levels=5 relzero;
run;
quit;
```

Example 9: Using GfK GeoMarketing Map Data to Produce a Simple Choropleth Map

Features: MAP= required argument referring to GfK map data

DATA= argument referring to response data

ID statement

CHORO statement options
- STAT=
- CDEFAULT=

Other features: System option FMTSEARCH=

SQL procedure

Data sets: MAPSGFK.EUROPE (map data)

DEMOGRAPHICS (table of response data)

Format: ison2a

Restriction: The GfK GeoMarketing map data set used in this example is licensed to be used only with SAS/GRAPH.

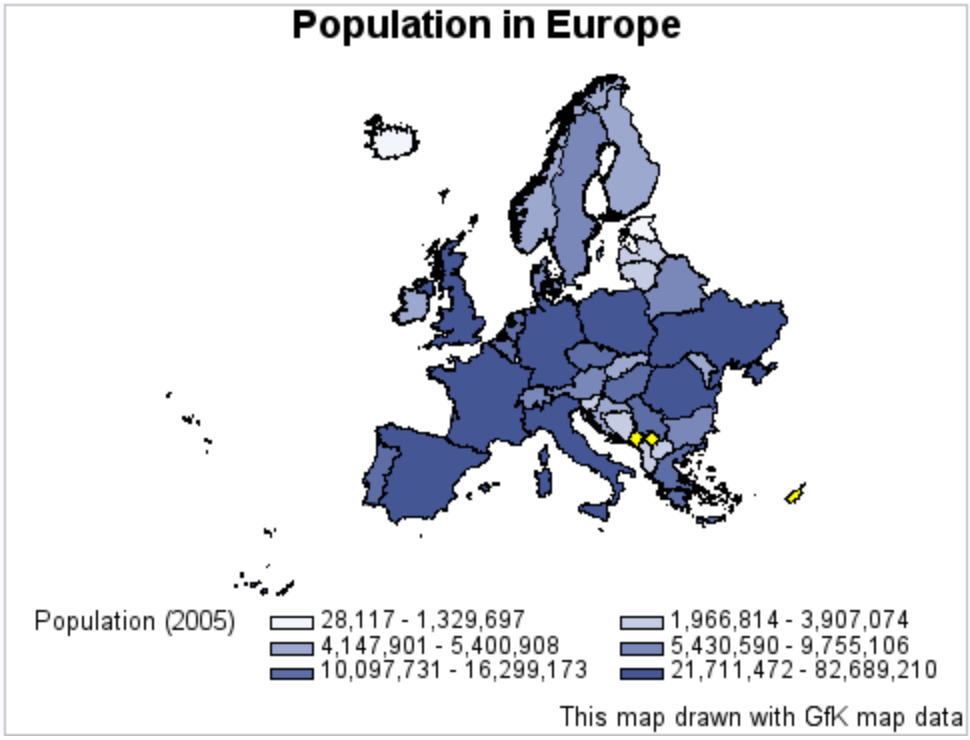
Sample library member: GMPGCHOR

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example produces a choropleth (two-dimensional) map that shows the population of countries in Europe. Because the DISCRETE option is not used, the response variable is assumed to have a continuous range of values. Because neither the LEVELS= nor MIDPOINTS= options are used, the GMAP procedure selects a number of levels based on the number of map areas. It then calculates the appropriate response levels. The legend shows the range of values for each level.



Program

```
options fmtsearch=(sashelp.mapfmts);

proc sql;
create table demographics(rename=(iso=oiso newiso=iso id=oldid newid=ID)) as
select demo.*,
put(demo.iso,z3.) as newiso format=$3.,
put(demo.iso,ison2a.) as newid
from sashelp.demographics as demo
;
alter table demographics
modify ID char(15) label='Alpha2 Country Code';
quit;

goptions reset=all border;

title1 "Population in Europe";
footnote j=r "This map drawn with GfK map data";

proc gmap map=mapsgfk.europe(where=(id not in:('SJ' 'TR')))
                data=demographics(where=(cont=93)) all;
id id;
choro pop / cdefault=yellow stat=first;
run;
quit;
```

Program Description

Specify the format catalog to search that has the predefined ISO alpha2 code.

```
options fmtsearch=(sashelp.mapfmts);
```

Create a table named demographics using sashelp.demographics as the base and changing its variables to match GfK variable types and lengths. This table is used as the response data set. Note that the ISO variable was numeric in the original sashelp.demographics data but is the character variable OISO in the GfK map data set. The format 'ison2a' uses the country's ISO numeric code to display the country's ISO alpha2 code. Also note that the ID variable was a numeric geographic locator code (glc) in the original sashelp.demographics data. However, it is represented by the ISOALPHA2 variable in the GfK map data set.

```
proc sql;
create table demographics(rename=(iso=oiso newiso=iso id=oldid newid=ID)) as
select demo.*,
put(demo.iso,z3.) as newiso format=$3.,
put(demo.iso,ison2a.) as newid
from sashelp.demographics as demo
;
alter table demographics
modify ID char(15) label='Alpha2 Country Code';
quit;
```

Set the graphics environment.

```
goptions reset=all border;
```

Define the title and the footnote for the map.

```
title1 "Population in Europe";
footnote j=r "This map drawn with GfK map data";
```

Produce the choropleth map. The ALL argument specifies that the output should include all of the map areas from the map data set. This happens even if the response data set DEMOGRAPHICS does not include an observation for the map area. The output shows three such areas. The ID statement specifies the variable that is in both the map data set and the response data set that defines map areas. CDEFAULT= specifies the color for the map areas that have missing data. In this map Montenegro, Kosovo, and the island of Cyprus are yellow, indicating missing data. The STATISTIC= option specifies that the GMAP procedure is to match the first observation from MAPSGFK.EUROPE data set and display the response value from this observation only. The WHERE= clause on the MAP= option excludes the islands of Cyprus and Svalbard, the country Turkey, and the Russian region. All of these map areas have no data in the DEMOGRAPHICS data set.

```
proc gmap map=mapsgfk.europe(where=(id not in:('SJ' 'TR')))
                data=demographics(where=(cont=93)) all;
id id;
choro pop / cdefault=yellow stat=first;
run;
quit;
```

Example 10: Using Traditional Map Data to Produce a Simple Choropleth Map

Features: MAP= required argument referring to traditional map data

DATA= required argument referring to response data

ID statement

CHORO statement option CDEFAULT=

Data sets: MAPS.EUROPE (map data)
SASHELP.DEMOGRAPHICS (response data)

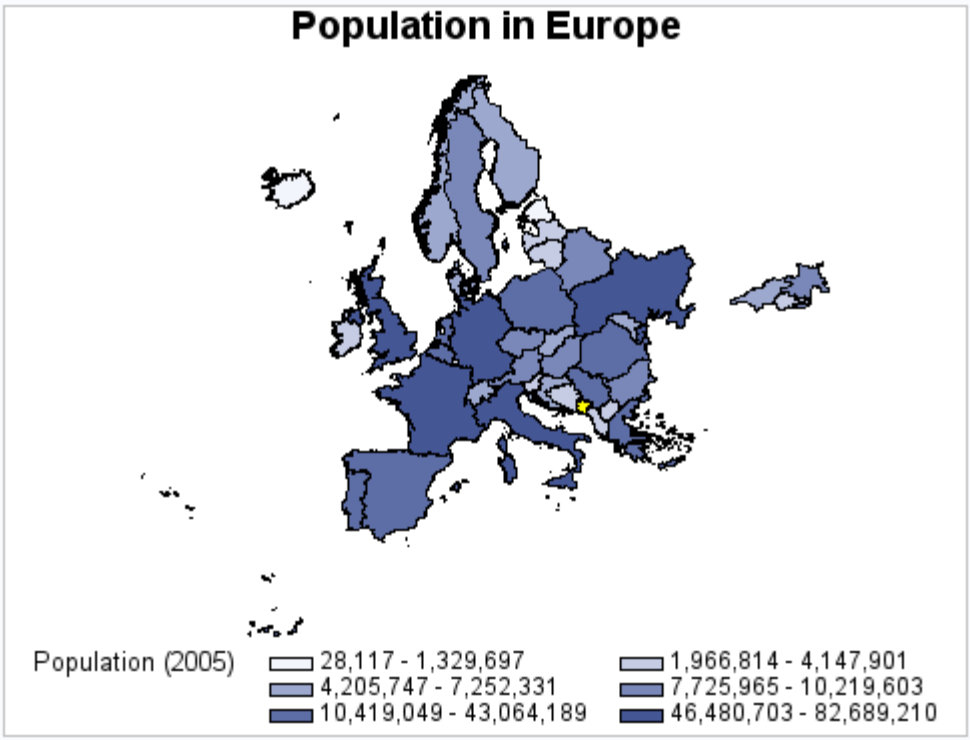
Sample library member: GMPCHORO

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example produces a choropleth (two-dimensional) map that shows the population of countries in Europe. Because the DISCRETE option is not used, the response variable is assumed to have a continuous range of values. Because neither the LEVELS= nor MIDPOINTS= options are used, the GMAP procedure selects a number of levels based on the number of map areas. It then calculates the appropriate response levels. The legend shows the range of values for each level.



Program

```
goptions reset=all border;

title1 "Population in Europe";

proc gmap map=maps.europe(where=(id ne 405 and id ne 845))
          data=sashelp.demographics(where=(cont=93)) all;
  id id;
  choro pop / cdefault=yellow;
```

```
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the title for the map.

```
title1 "Population in Europe";
```

Produce the choropleth map. The ALL argument specifies that the map displayed should include all of the map areas from the map data set. This happens even if the response data set DEMOGRAPHICS does not include an observation for the map area. The map shows one such area. The ID statement specifies the variable that is in both the map data set and the response data set that defines map areas. CDEFAULT= specifies the color for the map areas that have missing data. The WHERE= clause on the MAP= option excludes the islands of Greenland and Svalbard, which have no data in DEMOGRAPHICS data set. The WHERE= clause in the DATA= statement includes the Russian region countries Azerbaijan, Armenia, and Georgia.

```
proc gmap map=maps.europe(where=(id ne 405 and id ne 845))
          data=sashelp.demographics(where=(cont=93)) all;
  id id;
  choro pop / cdefault=yellow;
run;
quit;
```

Example 11: Using Traditional Map Data to Produce a Drilldown Choropleth Map

Features: MAP= required argument referring to traditional map data

DATA= required argument referring to response data

BY statement

ID statement

CHORO statement options
- LEVELS=
- NOLEGEND
- COUTLINE=
- HTML=
- DES=
- NAME=

Data sets: MAPSSAS.US (map data)

MYDATA (response data)

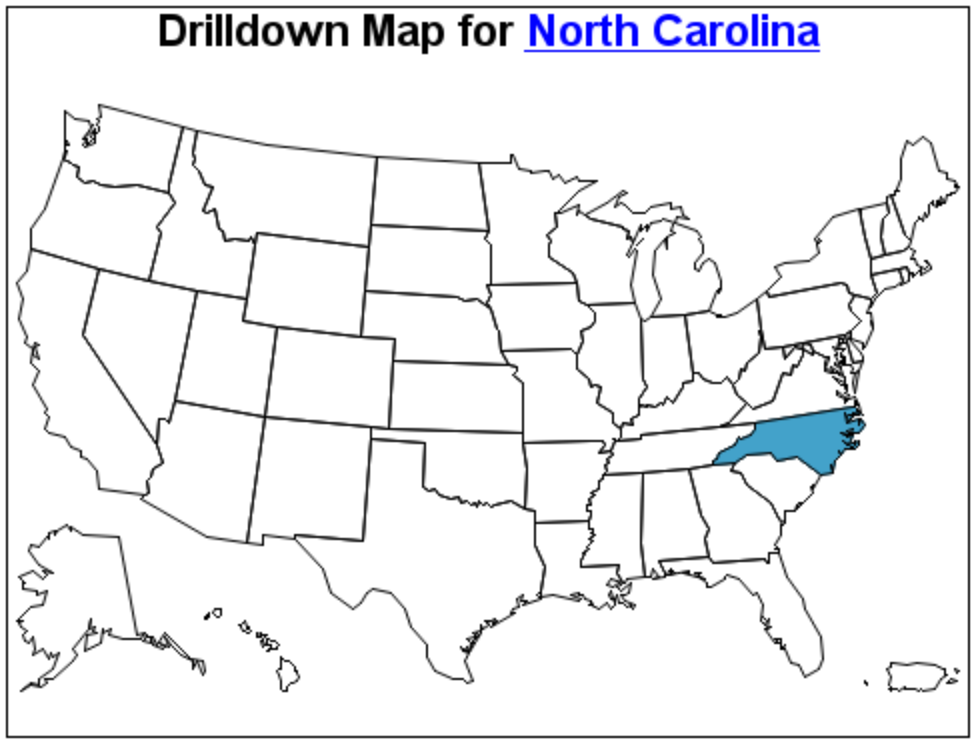
Sample library member: GMPCHORD

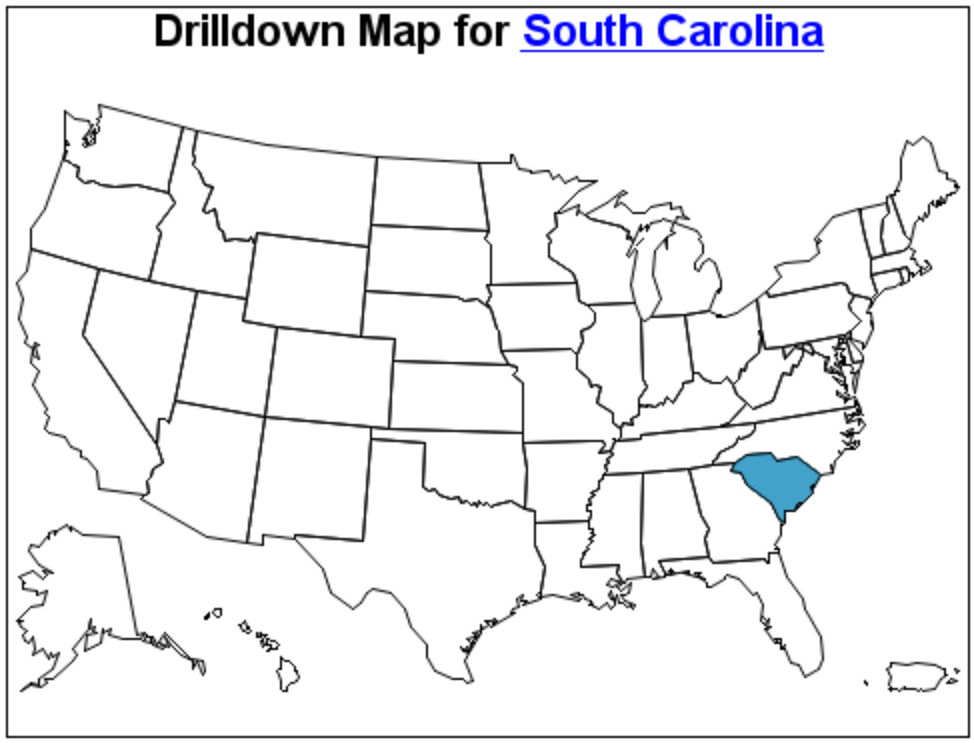
Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

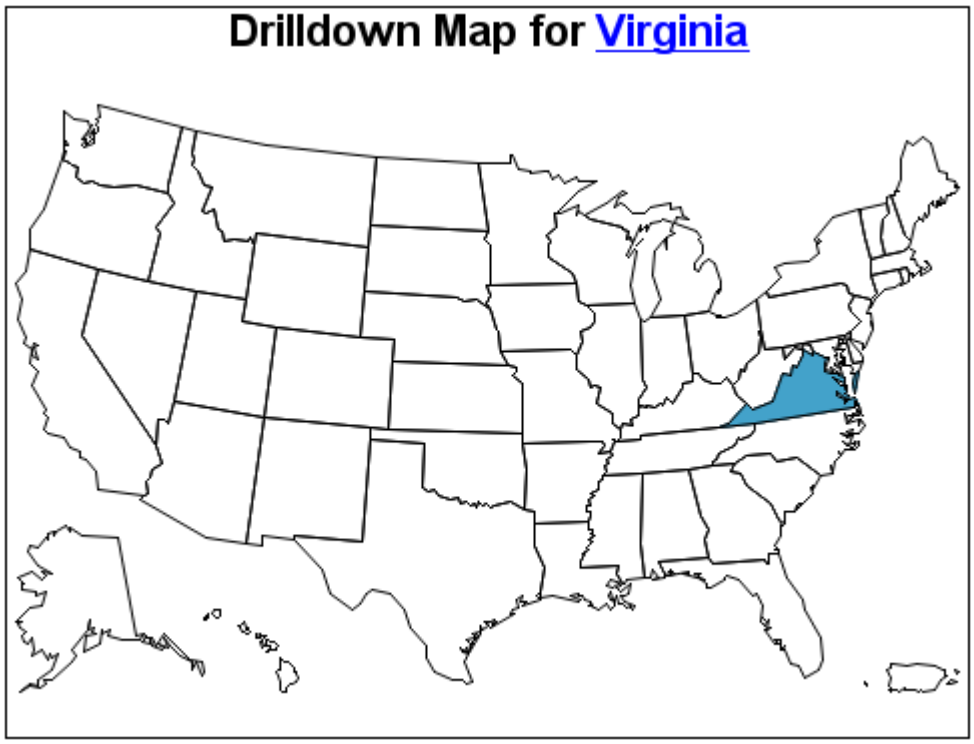
The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example produces three choropleth (two-dimensional) maps, each showing the population of a southeastern state in America. Each graph's title includes a link for drilling down to that particular state's home page on the web. The 'ALT=' option text description on each map title changes with a BY variable substitution to match the state being highlighted. The LEVELS= option enables the GMAP procedure to graph one response level for the population. This population number is part of the text description as your mouse pointer is held over the state colored in blue.







Program

```
%let name=map001;
filename odsout '.';

data mydata;
format population comma12.0;
input statecode $ 1-2 population;
```

```
year=2000;
statename=fipnamel(stfips(statecode));
datalines;
VA  7078515
NC  8049313
SC  4012012
;
run;

proc sort data=mydata out=mydata;
by statecode statename;
run;

data mydata; set mydata;
length htmlvar $500;
htmlvar='title='||quote(
 'State: '|| trim(left(statename)) ||'0D'x||
 'Population: '|| trim(left(put(population,comma12.0)))
 )||
 ' href="http://www.state.'||trim(left(lowcase(statecode)))||'.us"';
run;

goptions device=png gunit=pct htitle=6 htext=4 border;
ods _all_ close;
ods html path=odsout body="&name..htm" (title="SAS/Graph ODS HTML Drilldown Map")
 style=sasweb;

axis1 label=none;
axis2 label=none minor=none offset=(0,0);

pattern v=solid color=cx43a2ca;

options nobyline;
title1 "Drilldown Map for "
 color=blue underlin=1
 link="http://www.state.#byval(statecode)..us"
 alt="Click to drilldown to #byval(statename) home page"
 "#byval(statename)";

proc gmap data=mydata map=mapssas.us all;
by statecode statename;
id statecode;
choro population / levels=1 nolegend
 coutline=black
 html=htmlvar
 des=''
 name="&name._#byval(statecode)";
run;
quit;

ods html close;
ods html;
```

Program Description

Set up the output filename.

```
%let name=map001;
filename odsout '.';
```

Create the response data set MYDATA with the U.S. states to map.

```
data mydata;
format population comma12.0;
input statecode $ 1-2 population;
year=2000;
statename=fipnamel(stfips(statecode));
datalines;
VA  7078515
NC  8049313
SC  4012012
;
run;
```

Sort the response data.

```
proc sort data=mydata out=mydata;
by statecode statename;
run;
```

Set up the text when holding your mouse pointer over a highlighted state.

```
data mydata; set mydata;
length htmlvar $500;
htmlvar='title='||quote(
 'State: '|| trim(left(statename)) ||'0D'x||
 'Population: '|| trim(left(put(population,comma12.0)))
 )||
 ' href="http://www.state.'||trim(left(lowcase(statecode)))||'.us"';
run;
```

Set the graphics environment. The following ODS statements close all previously open destinations and store PNG output in your working directory. From there you can open the file in a web browser.

```
goptions device=png gunit=pct htitle=6 htext=4 border;
ods _all_ close;
ods html path=odsout body="&name..htm" (title="SAS/Graph ODS HTML Drilldown Map")
 style=sasweb;
```

Suppress the axis labels.

```
axis1 label=none;
axis2 label=none minor=none offset=(0,0);
```

Set the color to highlight the state.

```
pattern v=solid color=cx43a2ca;
```

Define the title and its drilldown link for the map. NOBYLINE suppresses the output of both BY variables indicated in the BY statement. The individual BY variables are used in the LINK= and ALT= option specifications. #BYVAL substitutes the current value of the BY variable, (statecode) or (statename), for #BYVAL in the text string and displays it.

```
options nobyline;
title1 "Drilldown Map for "
 color=blue underlin=1
```

```
 link="http://www.state.#byval(statecode)..us"
 alt="Click to drilldown to #byval(statename) home page"
 "#byval(statename)";
```

Produce the choropleth map. for each state. The BY statement specifies the variables in the response data set to work with. The ID statement specifies the FIPS state code variable that is in both the map data set and the response data set that defines map areas. The LEVELS= option enables the GMAP procedure to graph one response level for the population. COUTLINE= specifies that the map area be outlined in black. HTML= sets up the link that is displayed as your mouse pointer is held over the state in the output. NAME= defines the name of the GRSEG catalog entry and the name of the graphics output file.

```
proc gmap data=mydata map=mapssas.us all;
by statecode statename;
id statecode;
choro population / levels=1 nolegend
 coutline=black
 html=htmlvar
 des=''
 name="&name._#byval(statecode)";
run;
quit;
```

Reset the graphics environment. Run ODS HTML to close the output file, and then reopen ODS HTML. Reopening the HTML destination is not required when running the code in SAS Studio.

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

Example 12: Using GfK GeoMarketing Map Data When Labeling Provinces on a Map

Features: MAP= required argument referring to GfK map data

DATA= required argument referring to response data

ID statement

CHORO statement options
- ANNOTATE=
- NOLEGEND
- COUTLINE=
- STAT=

Other features: GREMOVE procedure

SORT procedure

Annotate Facility

Data set: MAPSGFK.BELARUS (map data)

Restriction: The GfK GeoMarketing map data set used in this example is licensed to be used only with SAS/GRAPH.

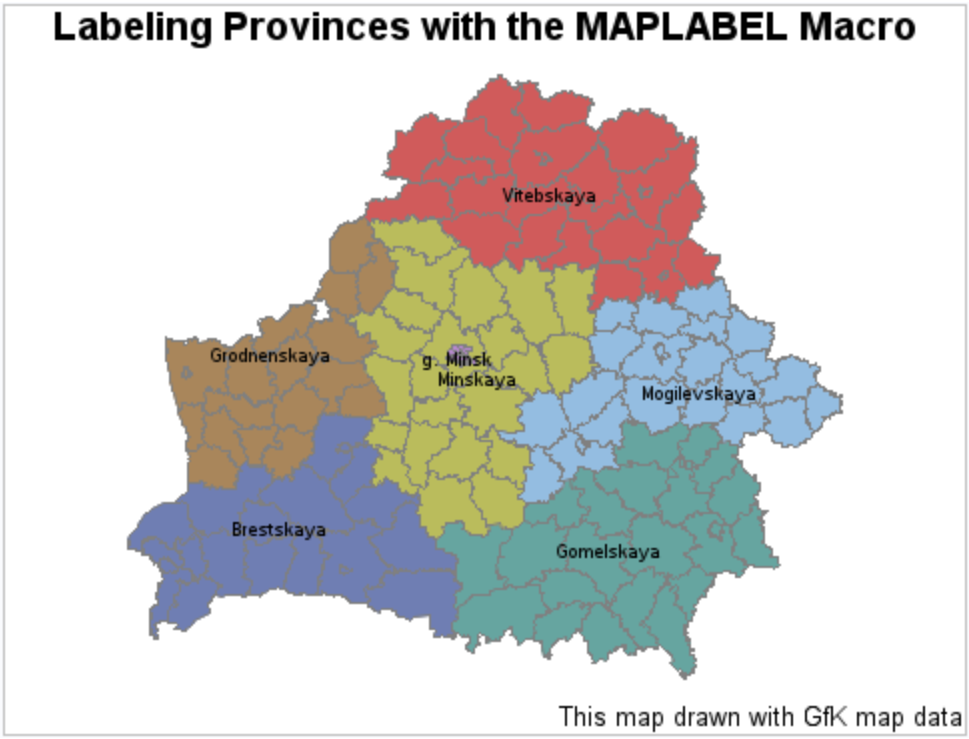
Sample library member: GMPGLABL

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example uses the Annotate facility to add labels to each area in a map of Belarus. The CHORO statement assigns the Annotate data set to the map. The %MAPLABEL Annotate macro is used to create and position the map labels. For more information about this macro, see "%MAPLABEL Macro" in *SAS/GRAPH: Reference*.



Program

```
goptions reset=all border;

title "Labeling Provinces with the MAPLABEL Macro";
footnote j=r "This map drawn with GfK map data";

proc sort data=mapsgfk.belarus out=belarus;
by id1 id;
run;

proc gremove data=belarus out=belarus_outline;
by id1;
id id;
run;

%annomac;
%maplabel (belarus_outline, mapsgfk.belarus_attr, work.labelout, id1name, id1,
           font=Albany AMT, color=black, size=2.5, hsys=3);
```

```
proc gmap map=belarus data=belarus;
   id id1 id;
   choro id1 / nolegend annotate=labelout coutline=gray stat=first;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the title and the footnote for the map.

```
title "Labeling Provinces with the MAPLABEL Macro";
footnote j=r "This map drawn with GfK map data";
```

Sort the GfK data by region and district variables.

```
proc sort data=mapsgfk.belarus out=belarus;
by id1 id;
run;
```

Use the GREMOVE procedure to remove the internal boundaries from the Belarus map. This prepares the data for the %MAPLABEL macro, which determines the centroid of each province and then center each province label.

```
proc gremove data=belarus out=belarus_outline;
by id1;
id id;
run;
```

Create the Annotate data set. The %ANNOMAC macro enables the Annotate macros. The %MAPLABEL Annotate macro creates the Annotate data set.

```
%annomac;
%maplabel (belarus_outline, mapsgfk.belarus_attr, work.labelout, id1name, id1,
           font=Albany AMT, color=black, size=2.5, hsys=3);
```

Produce the choropleth map. The NOLEGEND option suppresses the legend. The ANNOTATE= option specifies the data set to annotate the map. The COUTLINE= option outlines each province in gray. The STATISTIC= option specifies that the GMAP procedure match the first observation from the BELARUS data set and display the response value from this observation only.

```
proc gmap map=belarus data=belarus;
   id id1 id;
   choro id1 / nolegend annotate=labelout coutline=gray stat=first;
run;
quit;
```

Example 13: Using Traditional Map Data When Labeling Provinces on a Map

Features: MAP= required argument referring to traditional map data

DATA= required argument referring to response data

ID statement

CHORO statement options
- STAT=
- ANNOTATE=
- NOLEGEND

Other features: The global PATTERN statement

Annotate Facility

Data set: MAPS.BELARUS (map data)

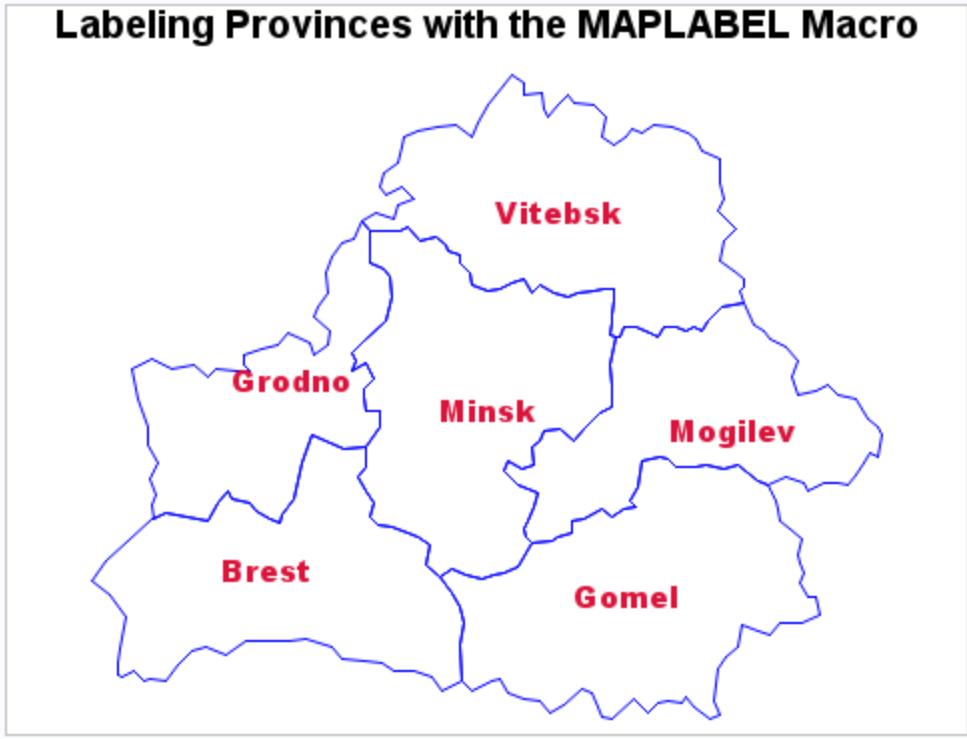
Sample library member: GMPLABEL

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example uses the Annotate facility to add labels to each area in a map of Belarus. The CHORO statement assigns the Annotate data set to the map. The %MAPLABEL Annotate macro is used to create and position the map labels. For more information about this macro, see "%MAPLABEL Macro" in *SAS/GRAPH: Reference*.



Program

```
goptions reset=all border;

title "Labeling Provinces with the MAPLABEL Macro";
```

```
pattern1 value=empty color=blue repeat=6;

%annomac;
%maplabel (maps.belarus, maps.belarus2, work.labelout, idname, id,
          font=Arial Black, color=crimson, size=4, hsys=3);

proc gmap map=maps.belarus data=maps.belarus;
   id id;
   choro id / stat=sum nolegend annotate=labelout;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the title for the map.

```
title "Labeling Provinces with the MAPLABEL Macro";
```

Define pattern characteristics. PATTERN1 defines a single map pattern that is repeated for each of the six map areas (provinces). The pattern is an empty fill with a blue border. The VALUE= option defines a map or plot pattern. Specifying a color causes PATTERN1 to generate only one pattern definition. The REPEAT= option specifies the number of times to repeat the pattern definition.

```
pattern1 value=empty color=blue repeat=6;
```

Create the Annotate data set. The %ANNOMAC macro enables the Annotate macros. The %MAPLABEL Annotate macro creates the Annotate data set.

```
%annomac;
%maplabel (maps.belarus, maps.belarus2, work.labelout, idname, id,
          font=Arial Black, color=crimson, size=4, hsys=3);
```

Produce the choropleth map. Use the option STATISTIC=SUM to chart the summed value of all observations with a given matching ID. The NOLEGEND option suppresses the legend. The ANNOTATE= option specifies the data set to annotate the map.

```
proc gmap map=maps.belarus data=maps.belarus;
   id id;
   choro id / stat=sum nolegend annotate=labelout;
run;
quit;
```

Example 14: Combining Traditional Map Data and Sample Response Data to Map U.S. Population Statistics

Features: MAP= required argument referring to traditional map data

DATA= required argument referring to response data in a sample data set

ID statement

CHORO statement options
- LEVELS=
- LEGEND=

Data sets: MAPSSAS.US (map data)
SASHELP.US_DATA (response data)

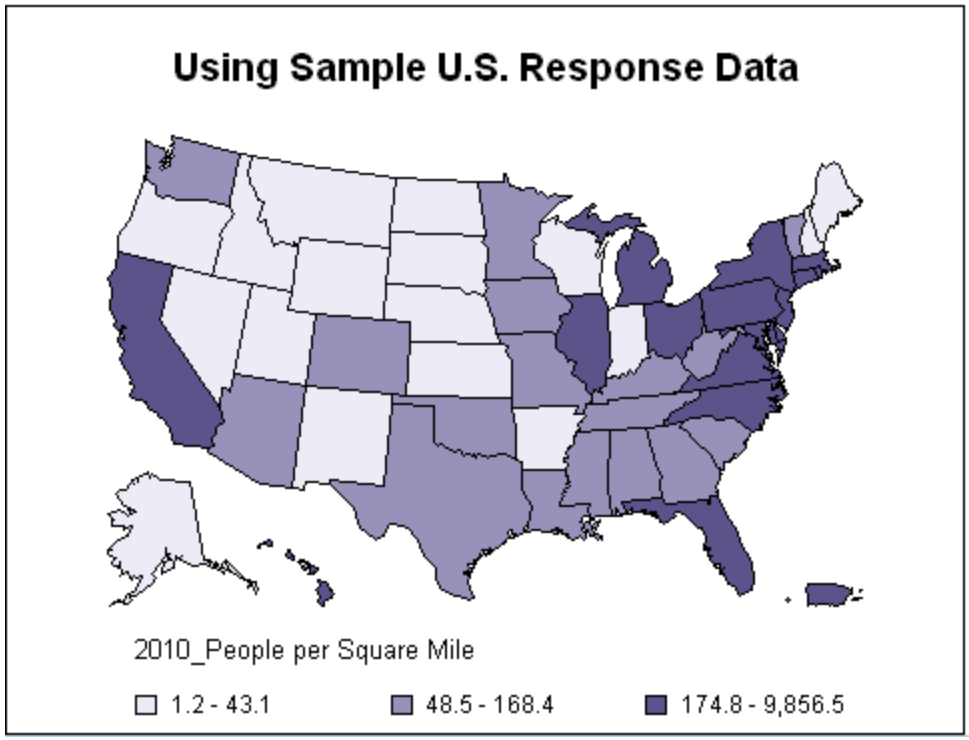
Sample library member: GMPUSDAT

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example uses a sample response data set named SASHELP.US_DATA that is specifically designed to be used with the traditional map data set MAPS.US or MAPSSAS.US. In this example, population details per state are mapped.



Program

```
goptions reset=all border;

title1 ls=1.5 "Using Sample U.S. Response Data";

legend1 label=(position=top) shape=bar(.1in,.1in);

proc gmap data=sashelp.us_data map=mapssas.us;
id state;
choro density_2010 / levels=3 legend=legend1;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the title for the map.

```
title1 ls=1.5 "Using Sample U.S. Response Data";
```

Define the legend for the map.

```
legend1 label=(position=top) shape=bar(.1in,.1in);
```

Produce the choropleth map of the United States. 2010 population densities are mapped per state. Use the sample response data set SASHELP.US_DATA, and the traditional map data set for the United States. The LEVELS= option specifies that 3 ranges of population values from the response data are graphed.

```
proc gmap data=sashelp.us_data map=mapssas.us;
id state;
choro density_2010 / levels=3 legend=legend1;
run;
quit;
```

Example 15: Using GfK GeoMarketing Map Data to Produce a Simple Prism Map

Features: MAP= required argument referring to GfK map data
DATA= argument referring to response data
ID statement
PRISM statement options
CDEFAULT=
RELZERO

Other features: System option FMTSEARCH=
SQL procedure

Data sets: MAPSGFK.AFRICA (map data)
DEMOGRAPHICS (table of response data)

Format: ison2a

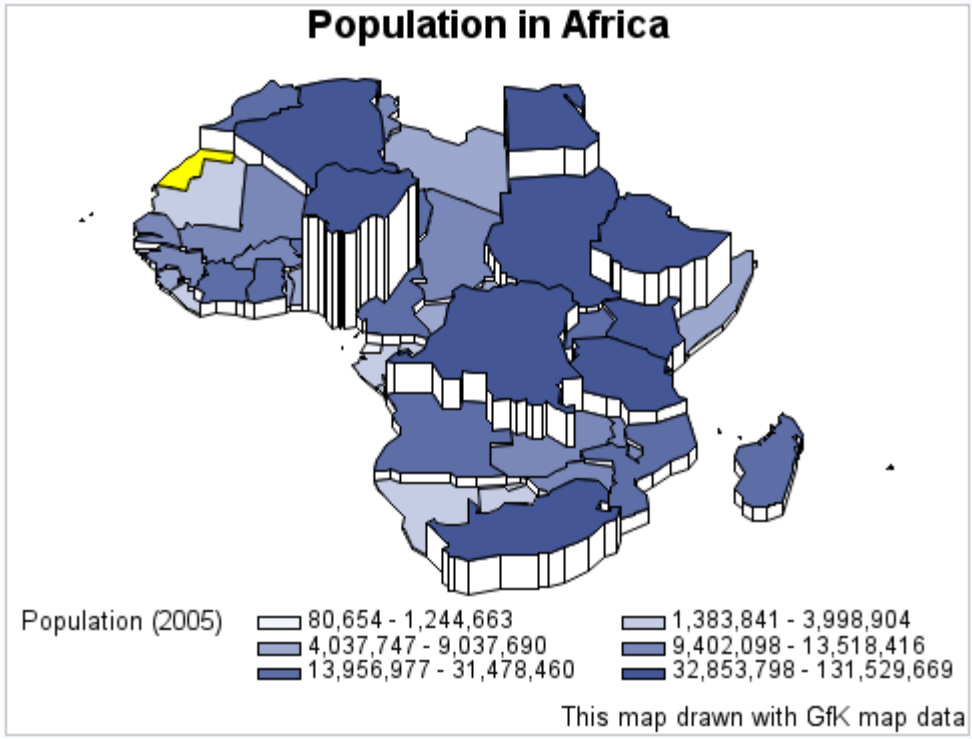
Restriction: The GfK GeoMarketing map data set used in this example is licensed to be used only with SAS/GRAPH.

Sample library member: GMPGPRSM

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output



Program

```
options fmtsearch=(sashelp.mapfmts);

proc sql;
create table demographics(rename=(iso=oiso newiso=iso id=oldid newid=ID)) as
select demo.*,
put(demo.iso,z3.) as newiso format=$3.,
put(demo.iso,ison2a.) as newid
from sashelp.demographics as demo
;
alter table demographics
modify ID char(15) label='Alpha2 Country Code';
quit;

goptions reset=all border;

title1 "Population in Africa";
footnote1 j=r "This map drawn with GfK map data";

proc gmap data=demographics(where=(cont=94))
          map=mapsgfk.africa(where=(lake=0)) density=low all;
   id id;
   prism pop / cdefault=yellow relzero;
run;
quit;
```

Program Description

Specify the format catalog to search that has the predefined ISO alpha2 code.

```
options fmtsearch=(sashelp.mapfmts);
```

Create a table named demographics using sashelp.demographics as the base and changing its variables to match GfK variable types and lengths. This table is used as the response data set. Note that the ISO variable was numeric in the original sashelp.demographics data but is the character variable OISO in the GfK map data set. The format 'ison2a' uses the country's ISO numeric code to display the country's ISO alpha2 code. Also note that the ID variable was a numeric geographic locator code (glc) in the original sashelp.demographics data. However, it is represented by the ISOALPHA2 variable in the GfK map data set.

```
proc sql;
create table demographics(rename=(iso=oiso newiso=iso id=oldid newid=ID)) as
select demo.*,
put(demo.iso,z3.) as newiso format=$3.,
put(demo.iso,ison2a.) as newid
from sashelp.demographics as demo
;
alter table demographics
modify ID char(15) label='Alpha2 Country Code';
quit;
```

Set the graphics environment.

```
goptions reset=all border;
```

Define the title and footnote for the map

```
title1 "Population in Africa";
footnote1 j=r "This map drawn with GfK map data";
```

Produce the prism map. The WHERE= clause in the DATA= statement includes the countries that are part of the continent of Africa. The WHERE= clause in the MAP= statement excludes lake areas. The ALL argument specifies that the map displayed should include all of the map areas from the map data set. This happens even if the response data set DEMOGRAPHICS does not include an observation for the map area. The map shows one such area. The ID statement specifies the variable in the map data set and the response data set that defines map areas. The CDEFAULT= option sets the color of map areas that have missing data. The RELZERO option makes the prism heights relative to zero.

```
proc gmap data=demographics(where=(cont=94))
          map=mapsgfk.africa(where=(lake=0)) density=low all;
   id id;
   prism pop / cdefault=yellow relzero;
run;
quit;
```

Example 16: Using Traditional Map Data to Produce a Simple Prism Map

Features: MAP= required argument referring to traditional map data

DATA= required argument referring to response data

ID statement

PRISM statement options

- CDEFAULT=
- RELZERO

Other features: SAS DATA step with assignment statements

Data sets: MAPS.AFRICA (map data)
SASHELP.DEMOGRAPHICS (response data)

Sample library member: GMPPRISM

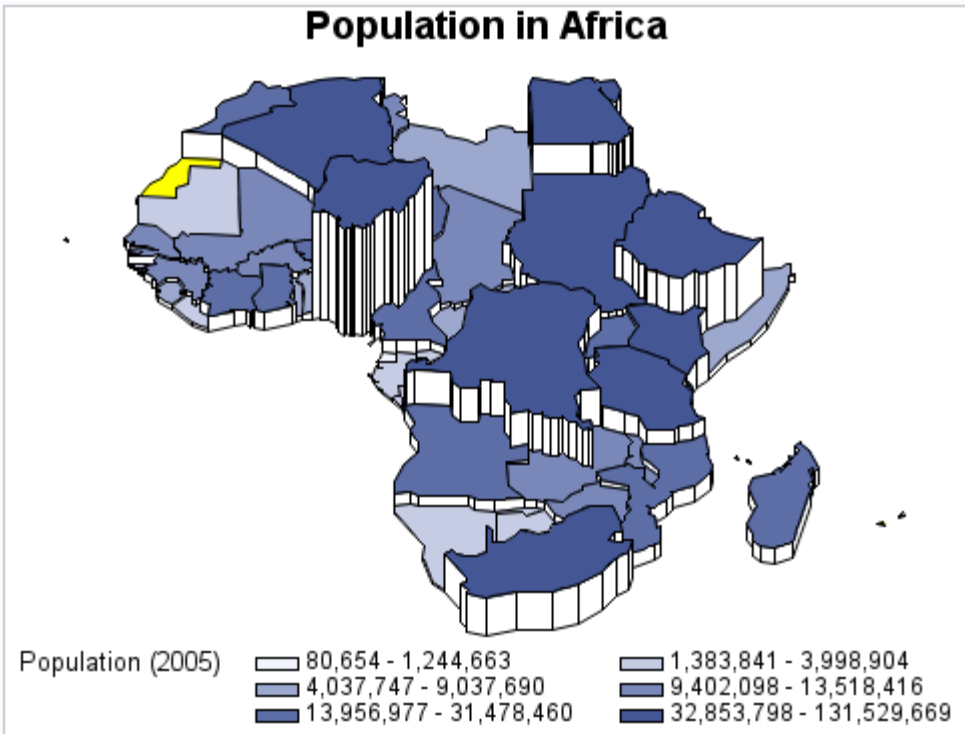
Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

This example produces a prism map of the population of countries in Africa. Because 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. It then calculates the appropriate response levels.

Because 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. Because 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.

Output



Program

```
goptions reset=all border;

title1 "Population in Africa";
```

```
data africa;
set maps.africa;
by id segment;
if first.id then lake=0;
if x=. then lake+1;
retain lake;
run;

proc gmap data=sashelp.demographics(where=(cont=94))
          map=africa(where=(lake=0)) density=low all;
   id id;
   prism pop / cdefault=yellow relzero;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the title for the map.

```
title1 "Population in Africa";
```

Identify lake regions in the map data.

```
data africa;
set maps.africa;
by id segment;
if first.id then lake=0;
if x=. then lake+1;
retain lake;
run;
```

Produce the prism map. The WHERE= clause in the MAP= statement excludes lake regions from the map. The ALL argument specifies that the map displayed should include all of the map areas from the map data set. This happens even if the response data set DEMOGRAPHICS does not include an observation for the map area. The map shows one such area. The ID statement specifies the variable in the map data set and the response data set that defines map areas. The CDEFAULT= option sets the color of map areas that have missing data. The RELZERO option makes the prism heights relative to zero.

```
proc gmap data=sashelp.demographics(where=(cont=94))
          map=africa(where=(lake=0)) density=low all;
   id id;
   prism pop / cdefault=yellow relzero;
run;
quit;
```

Example 17: Using GfK GeoMarketing Map Data When Specifying Midpoints in a Prism Map

Features: MAP= required argument referring to GfK map data

DATA= argument referring to response data

ID statement

FORMAT statement

PRISM statement options
- MIDPOINTS=
- CDEFAULT=

Other features: System option FMTSEARCH=

SQL procedure

PERCENTN format

Data sets: MAPSGFK.AFRICA (map data)

DEMOGRAPHICS (table of response data)

Format: ison2a

Restriction: The GfK GeoMarketing map data set used in this example is licensed to be used only with SAS/GRAPH.

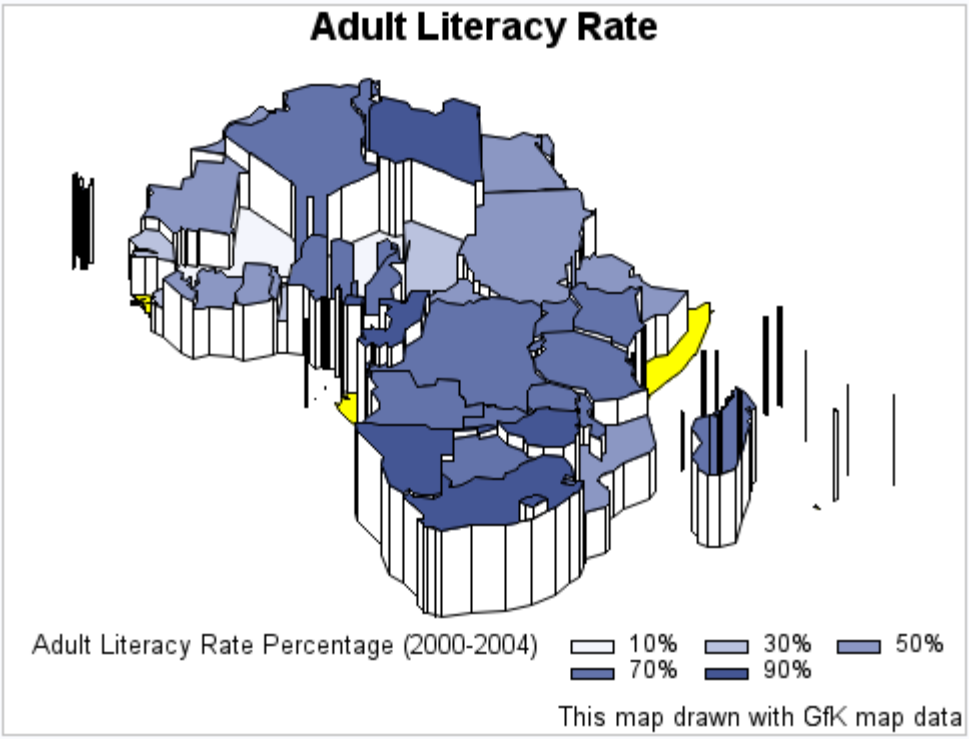
Sample library member: GMPGMDPT

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example specifies a set of midpoints that are used to create the response levels.



Program

```
options fmtsearch=(sashelp.mapfmts);

proc sql;
create table demographics(rename=(iso=oiso newiso=iso id=oldid newid=ID)) as
select demo.*,
put(demo.iso,z3.) as newiso format=$3.,
put(demo.iso,ison2a.) as newid
from sashelp.demographics as demo
;
alter table demographics
modify ID char(15) label='Alpha2 Country Code';
quit;

goptions reset=all border;

title1 "Adult Literacy Rate";
footnote1 j=r "This map drawn with GfK map data";

proc gmap data=demographics(where=(cont=94))
map=mapsgfk.africa(where=(lake=0)) density=low all;
id id;
format adultliteracypct percentn5.0;
prism adultliteracypct / midpoints=.10 to .90 by .20
                         cdefault=yellow;
run;
quit;
```

Program Description

Specify the format catalog to search that has the predefined ISO alpha2 code.

```
options fmtsearch=(sashelp.mapfmts);
```

Create a table named demographics using sashelp.demographics as the base and changing its variables to match GfK variable types and lengths. This table is used as the response data set. Note that the ISO variable was numeric in the original sashelp.demographics data but is the character variable OISO in the GfK map data set. The format 'ison2a' uses the country's ISO numeric code to display the country's ISO alpha2 code. Also note that the ID variable was a numeric geographic locator code (glc) in the original sashelp.demographics data. However, it is represented by the ISOALPHA2 variable in the GfK map data set.

```
proc sql;
create table demographics(rename=(iso=oiso newiso=iso id=oldid newid=ID)) as
select demo.*,
put(demo.iso,z3.) as newiso format=$3.,
put(demo.iso,ison2a.) as newid
from sashelp.demographics as demo
;
alter table demographics
modify ID char(15) label='Alpha2 Country Code';
quit;
```

Set the graphics environment.

```
goptions reset=all border;
```

Define the title and footnote for the map.

```
title1 "Adult Literacy Rate";
footnote1 j=r "This map drawn with GfK map data";
```

Produce the prism map. The WHERE= clause in the MAP= statement excludes lake regions from the map. The ALL argument specifies that the map displayed should include all of the map areas from the map data set. This happens even if the response data set DEMOGRAPHICS does not include an observation for the map area. The map shows three such areas. The MIDPOINTS= option specifies the response levels for the map. The CDEFAULT= option sets the color of map areas that have missing data.

```
proc gmap data=demographics(where=(cont=94))
map=mapsgfk.africa(where=(lake=0)) density=low all;
id id;
format adultliteracypct percentn5.0;
prism adultliteracypct / midpoints=.10 to .90 by .20
                         cdefault=yellow;
run;
quit;
```

Example 18: Using Traditional Map Data When Specifying Midpoints in a Prism Map

Features: MAP= required argument referring to traditional map data
DATA= required argument referring to response data
ID statement
FORMAT statement
PRISM statement options
MIDPOINTS=
CDEFAULT=

Other features: SAS DATA step with assignment statements
PERCENTN format

Data sets: MAPS.AFRICA (map data)
SASHELP.DEMOGRAPHICS (response data)

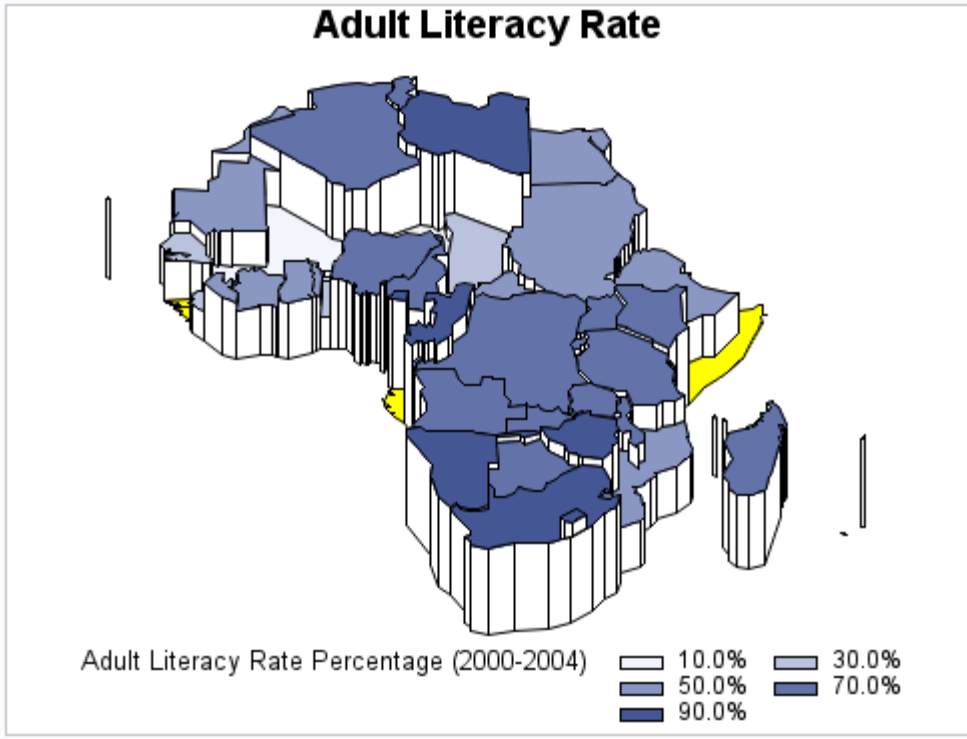
Sample library member: GMPMIDPT

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example specifies a set of midpoints that are used to create the response levels.



Program

```
goptions reset=all border;

title1 "Adult Literacy Rate";

data africa;
set maps.africa;
by id segment;
if first.id then lake=0;
if x=. then lake+1;
retain lake;
run;

proc gmap data=sashelp.demographics(where=(cont=94))
          map=africa(where=(lake=0)) density=low all;
   id id;
   format adultliteracypct percentn5.0;
   prism adultliteracypct / midpoints=.1 to .9 by .2
                            cdefault=yellow;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the title for the map.

```
title1 "Adult Literacy Rate";
```

Identify lake regions in the map data.

```
data africa;
set maps.africa;
by id segment;
if first.id then lake=0;
if x=. then lake+1;
retain lake;
run;
```

Produce the prism map. The ALL argument specifies that the map displayed should include all of the map areas from the map data set. This happens even if the response data set DEMOGRAPHICS does not include an observation for the map area. The map shows three such areas. The MIDPOINTS= option specifies the response levels for the map. The CDEFAULT= option sets the color of map areas that have missing data.

```
proc gmap data=sashelp.demographics(where=(cont=94))
          map=africa(where=(lake=0)) density=low all;
   id id;
   format adultliteracypct percentn5.0;
   prism adultliteracypct / midpoints=.1 to .9 by .2
                            cdefault=yellow;
run;
quit;
```

Example 19: Using GfK GeoMarketing Map Data to Produce a Simple Surface Map

Features: MAP= required argument referring to GfK map data
DATA= argument referring to response data
ID statement
SURFACE statement option STAT=

Other features: System option FMTSEARCH=
SQL procedure

Data sets: MAPSGFK.SAMERICA (map data)
DEMOGRAPHICS (table of response data)

Format: ison2a

Restriction: The GfK GeoMarketing map data set used in this example is licensed to be used only with SAS/GRAPH.

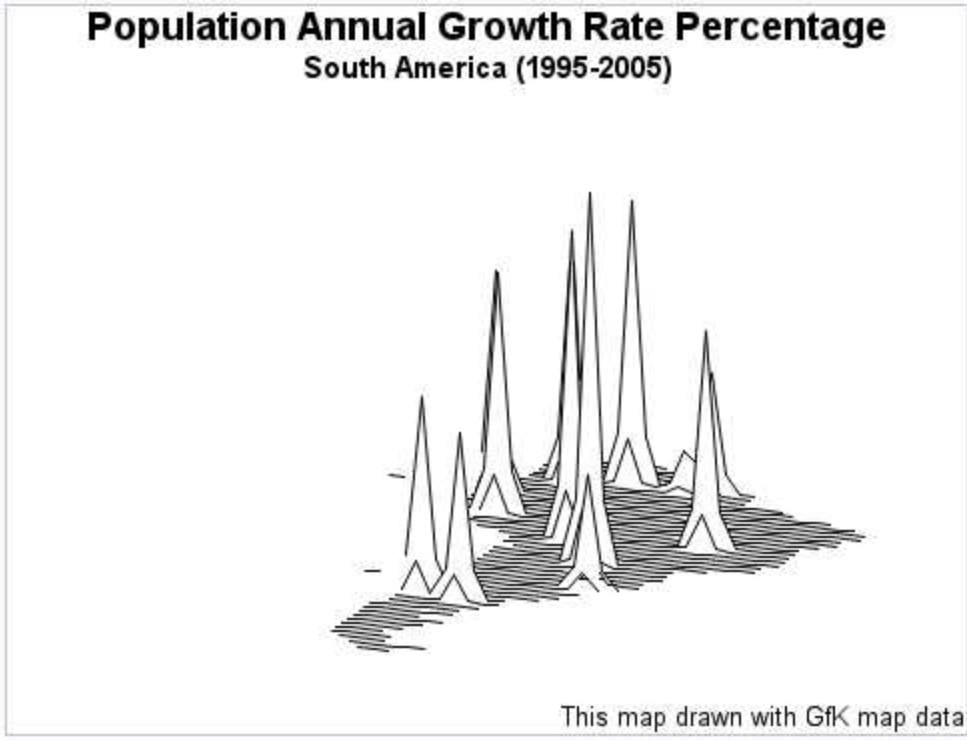
Sample library member: GMPGSURF

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example produces a surface map that shows the annual population growth rate of countries in South America. Not all the countries that are represented in the GfK map data set have demographic data in the response data set. The CONSTANT= and NLINES= options are not used. Therefore, 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.



Program

```
options fmtsearch=(sashelp.mapfmts);

proc sql;
create table demographics(rename=(iso=oiso newiso=iso id=oldid newid=ID)) as
select demo.*,
put(demo.iso,z3.) as newiso format=$3.,
put(demo.iso,ison2a.) as newid
from sashelp.demographics as demo
;
alter table demographics
modify ID char(15) label='Alpha2 Country Code';
quit;

goptions reset=all border;

title1 "Population Annual Growth Rate Percentage";
title2 "South America (1995-2005)";
footnote1 j=r "This map drawn with GfK map data";

proc gmap map=mapsgfk.samerica data=demographics;
id id;
surface popagr / stat=sum;
```

```
run;
quit;
```

Program Description

Specify the format catalog to search that has the predefined ISO alpha2 code.

```
options fmtsearch=(sashelp.mapfmts);
```

Create a table named demographics using sashelp.demographics as the base and changing its variables to match GfK variable types and lengths. This table is used as the response data set. Note that the ISO variable was numeric in the original sashelp.demographics data but is the character variable OISO in the GfK map data set. The format 'ison2a' uses the country's ISO numeric code to display the country's ISO alpha2 code. Also note that the ID variable was a numeric geographic locator code (GLC) in the original sashelp.demographics data. However, it is represented by the ISOALPHA2 variable in the GfK map data set.

```
proc sql;
create table demographics(rename=(iso=oiso newiso=iso id=oldid newid=ID)) as
select demo.*,
put(demo.iso,z3.) as newiso format=$3.,
put(demo.iso,ison2a.) as newid
from sashelp.demographics as demo
;
alter table demographics
modify ID char(15) label='Alpha2 Country Code';
quit;
```

Set the graphics environment.

```
goptions reset=all border;
```

Define the titles and the footnote for the map

```
title1 "Population Annual Growth Rate Percentage";
title2 "South America (1995-2005)";
footnote1 j=r "This map drawn with GfK map data";
```

Produce the surface map. Use the option STATISTIC=SUM to chart the summed value of all observations with a given matching ID. The ID statement specifies the variable in the map data set and the response data set that defines the map areas.

```
proc gmap map=mapsgfk.samerica data=demographics;
id id;
surface popagr / stat=sum;
run;
quit;
```

Example 20: Using Traditional Map Data to Produce a Simple Surface Map

Features: MAP= required argument referring to traditional map data

DATA= required argument referring to response data

ID statement

SURFACE statement

Data sets: MAPSSAS.SAMERICA (map data)
SASHELP.DEMOGRAPHICS (response data)

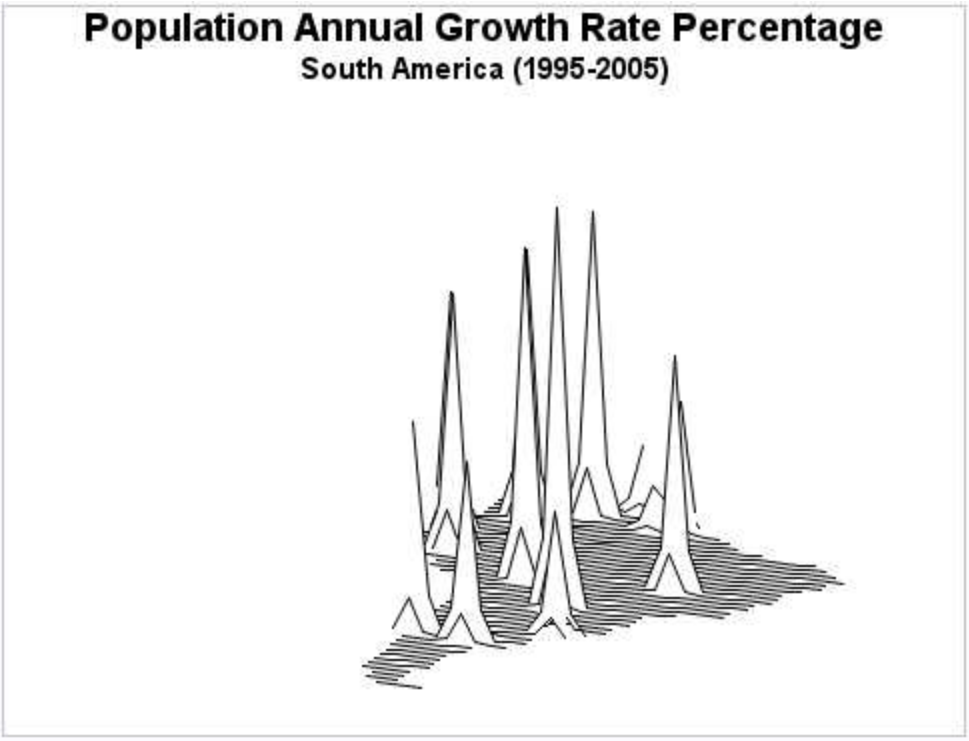
Sample library member: GMPSURFA

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example produces a surface map that shows the annual population growth rate of countries in South America. Not all the countries that are represented in the traditional map data set have demographic data in the response data set. The CONSTANT= and NLINES= options are not used. Therefore, 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.



Program

```
goptions reset=all border;

title1 "Population Annual Growth Rate Percentage";
title2 "South America (1995-2005)";

proc gmap map=mapssas.samerica data=sashelp.demographics;
   id id;
```

```
   surface popagr;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the title for the map.

```
title1 "Population Annual Growth Rate Percentage";
title2 "South America (1995-2005)";
```

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=mapssas.samerica data=sashelp.demographics;
   id id;
   surface popagr;
run;
quit;
```

Example 21: Mapping an Individual Country By Subsetting MAPS.WORLD

Features: OPTIONS statement
MAPS= required argument referring to traditional map data
DATA= required argument referring to response data
FORMAT statement
ID statement
CHORO statement option STAT=

Other features: SQL procedure

Data sets: MAPS.WORLD (map data)
MYDATA (table with subsetted map data)

Format: GLCNSM

Sample library member: GMPCOUNT

Notes: This example cannot be run with a GfK map data set. The GLC codes are replaced by ISO codes and are not supported for GfK mapping.

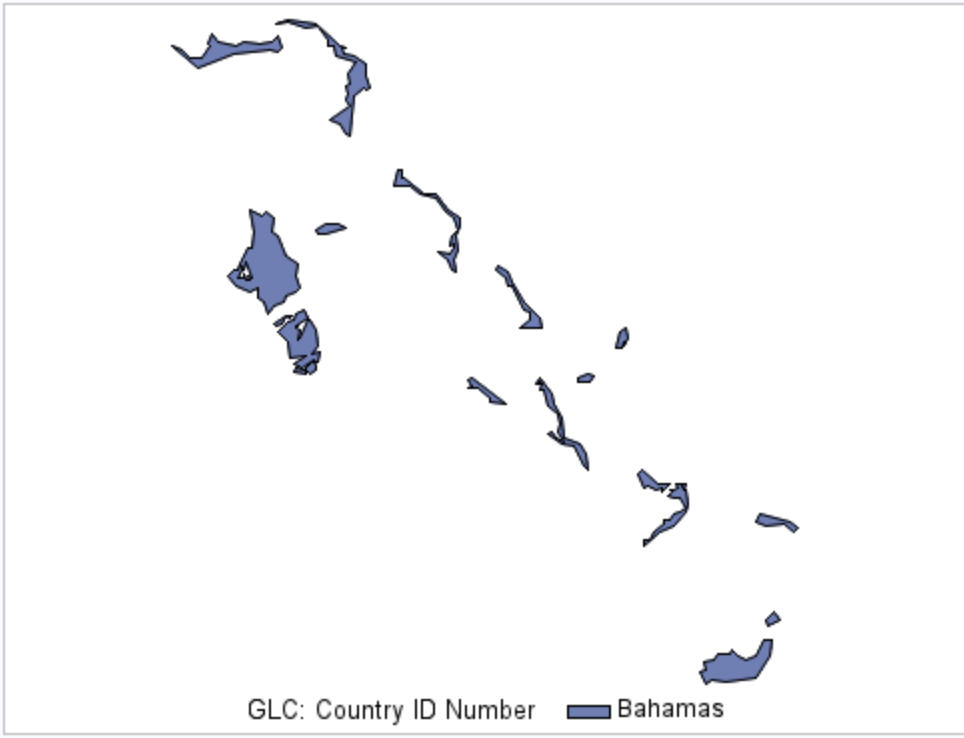
The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example shows how to subset a single country from the MAPS.WORLD data set. This is useful for mapping smaller countries that do not have an individual map data set in the MAPS or MAPSSAS library.

This example also demonstrates how to use the GLCNSM. format to specify a country by name rather than by its ID value.



Program

```
goptions reset=all border;

options fmtsearch=(sashelp.mapfmts);

proc sql;
 create table mymap as
 select * from maps.world
 where put(id,glcnsm.) eq "Bahamas"
 ;
 run;

proc gmap map=mymap data=mymap;
 format id glcnsm.;
 id id;
 choro id / stat=first;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Enable the formats from the SASHELP.MAPFMTS catalog.

```
options fmtsearch=(sashelp.mapfmts);
```

Subset the map data for the Bahamas from MAPS.WORLD by using the SQL procedure and the geographic locator code format (GLCNSM).

```
proc sql;
 create table mymap as
 select * from maps.world
 where put(id,glcnsm.) eq "Bahamas"
 ;
 run;
```

Create the map by using the GMAP procedure. The STATISTIC= option specifies that the GMAP procedure is to match the first observation from the MYMAP data set and display the response value from this observation only.

```
proc gmap map=mymap data=mymap;
 format id glcnsm.;
 id id;
 choro id / stat=first;
run;
quit;
```

Example 22: Using GfK GeoMarketing Map Data to Specify Country Subdivisions on a Continent

Features: MAP= required argument referring to GfK map data

DATA= argument referring to response data

ID statement

CHORO statement options
- STATISTIC=
- DISCRETE
- NOLEGEND

Data set: MAPSGFK.AFRICA1 (map and response data)

Restriction: The GfK GeoMarketing map data set used in this example is licensed to be used only with SAS/GRAPH.

Sample library member: GMPGCON1

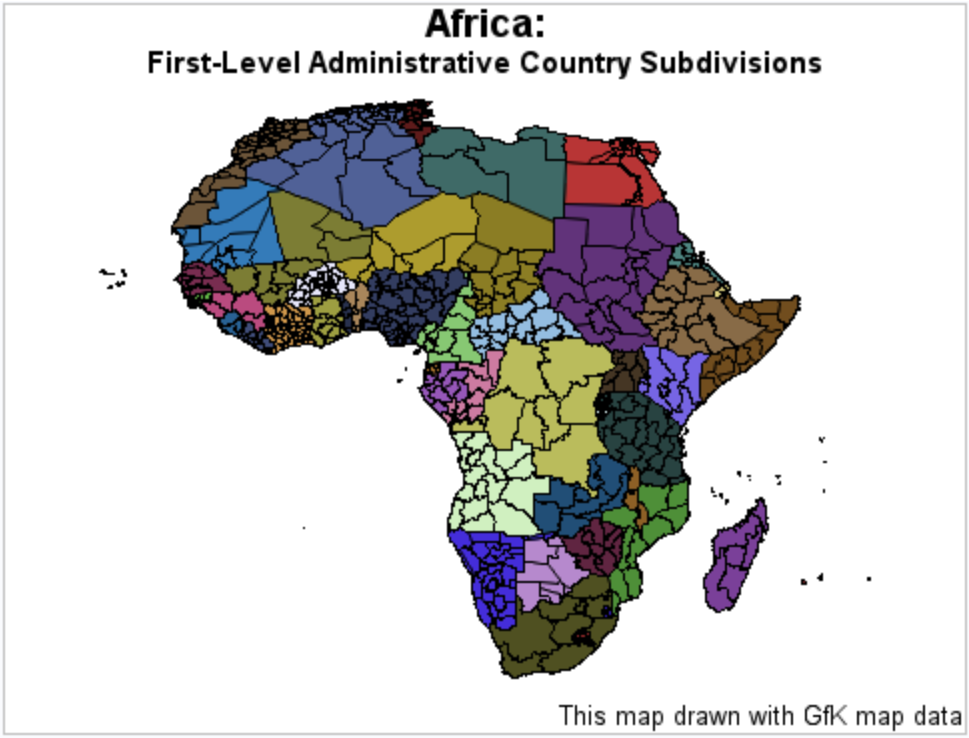
Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example uses the first-level administrative detail in countries offering that detail in the GfK map data set to show the subdivisions within countries in the African continent.

This level of detail is available only by country and not at a continent level with the traditional map data sets.



Program

```
goptions reset=all border;

title1 j=c 'Africa:';
title2 j=c 'First-Level Administrative Country Subdivisions';
footnote1 j=r 'This map drawn with GfK map data';

pattern1 value=msolid color=CXD0F0C0;
pattern2 value=ms color=CXE6E6FA;
pattern3 value=ms color=CXE7FEFF;

proc gmap map=mapsgfk.africa1 data=mapsgfk.africa1;
id isoalpha2 id;
choro isoalpha2/discrete nolegend stat=first;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the titles and footnote for the map.

```
title1 j=c 'Africa:';
title2 j=c 'First-Level Administrative Country Subdivisions';
footnote1 j=r 'This map drawn with GfK map data';
```

Change the color in some indistinguishable areas.

```
pattern1 value=msolid color=CXD0F0C0;
pattern2 value=ms color=CXE6E6FA;
pattern3 value=ms color=CXE7FEFF;
```

Produce the choropleth map. The DISCRETE= option generates a separate color for each different response variable value. The NOLEGEND= option specifies that no legend statement is associated with the map. The STATISTIC= option specifies that the GMAP procedure is to match the first observation from MAPSGFK.AFRICA1 data set and display the response value from this observation only.

```
proc gmap map=mapsgfk.africa1 data=mapsgfk.africa1;
id isoalpha2 id;
choro isoalpha2/discrete nolegend stat=first;
run;
quit;
```

Example 23: Using GfK GeoMarketing Map Data to Produce a Choropleth Map Combining Three Map Data Sets

Features: MAP= required argument referring to GfK map data

DATA= argument referring to response data

ID statement

CHORO statement options
- STATISTIC=
- DISCRETE
- LEGEND

Other features: SAS DATA step with assignment statements

GPROJECT procedure

Data sets: MAPSGFK.AUSTRIA (map data)

MAPSGFK.CZECH_REPUBLIC (map data)

MAPSGFK.HUNGARY (map data)

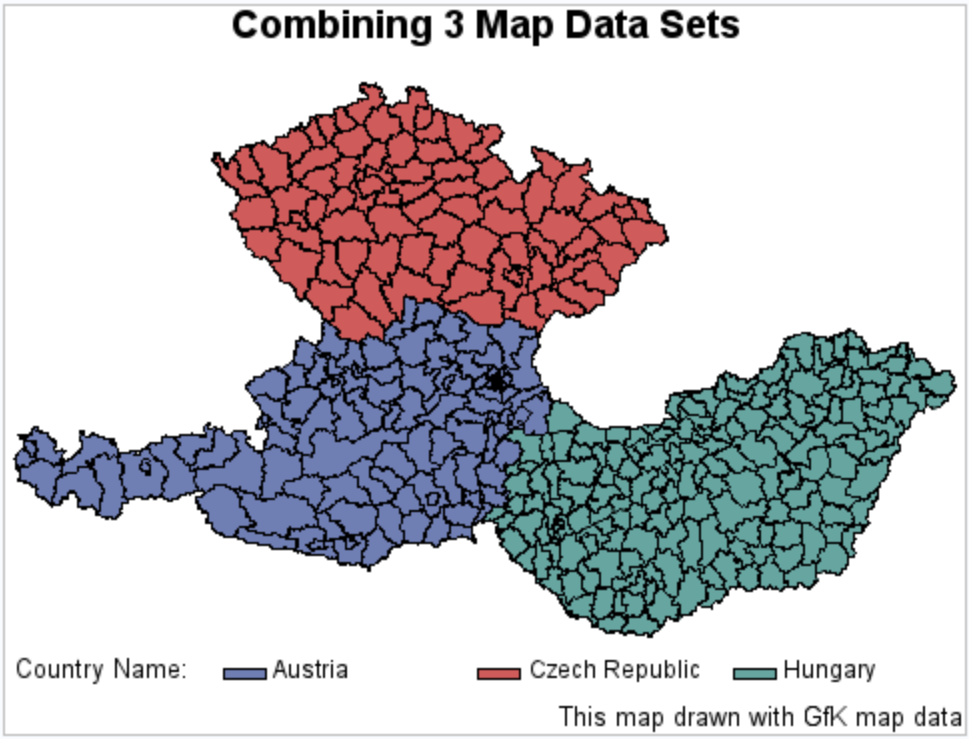
Restriction: The GfK GeoMarketing map data set used in this example is licensed to be used only with SAS/GRAPH.

Sample library member: GMPGCONC

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output



Program

```
data all;
length name $18;
set mapsgfk.austria(in=k1) mapsgfk.czech_republic(in=k2) mapsgfk.hungary;

if k1 then name='Austria';
else if k2 then name='Czech Republic';
else name='Hungary';
x=long;
y=lat;
run;

proc gproject data=all out=map degrees eastlong parmout;
id name id;
run;

goptions reset=all border;

title1 "Combining 3 Map Data Sets";
footnote j=r "This map drawn with GfK map data";
legend1 label=("Country Name:");

proc gmap map=map data=map;
id name id;
choro name/discrete stat=first legend=legend1;
run;
quit;
```

Program Description

Specify the countries to combine.

```
data all;
length name $18;
set mapsgfk.austria(in=k1) mapsgfk.czech_republic(in=k2) mapsgfk.hungary;

if k1 then name='Austria';
else if k2 then name='Czech Republic';
else name='Hungary';
x=long;
y=lat;
run;
```

Project the map and store the projection parameters.

```
proc gproject data=all out=map degrees eastlong parmout;
id name id;
run;
```

Set the graphics environment.

```
goptions reset=all border;
```

Define the title, footnote, and legend for the map. The LEGEND= statement relabels the NAME variable.

```
title1 "Combining 3 Map Data Sets";
footnote j=r "This map drawn with GfK map data";
legend1 label=("Country Name:");
```

Produce the choropleth map that combines 3 European countries. The DISCRETE= option generates a separate color for each different response variable value. The STATISTIC= option specifies that the GMAP procedure is to match the first observation from each of the three MAPSGFK data sets. It displays the response value from that observation only. The LEGEND= option pulls in the LEGEND statement's label assignment.

```
proc gmap map=map data=map;
id name id;
choro name/discrete stat=first legend=legend1;
run;
quit;
```

Example 24: Using GfK GeoMarketing Map Data to Produce a Choropleth Map Annotating Cities

Features: MAP= required argument referring to GfK map data

DATA= argument referring to response data

ID statement

CHORO statement options

- DISCRETE
- XSIZE=
- COUTLINE=

ANNOTATE=
DESCRIPTION=
NAME=
STATISTIC=
LEGEND=

Other features: SAS DATA step with assignment statements
GPROJECT procedure
Annotate facility

Data sets: MAPSGFK.AUSTRIA (map data)
MAPSGFK.CZECH_REPUBLIC (map data)
MAPSGFK.HUNGARY (map data)

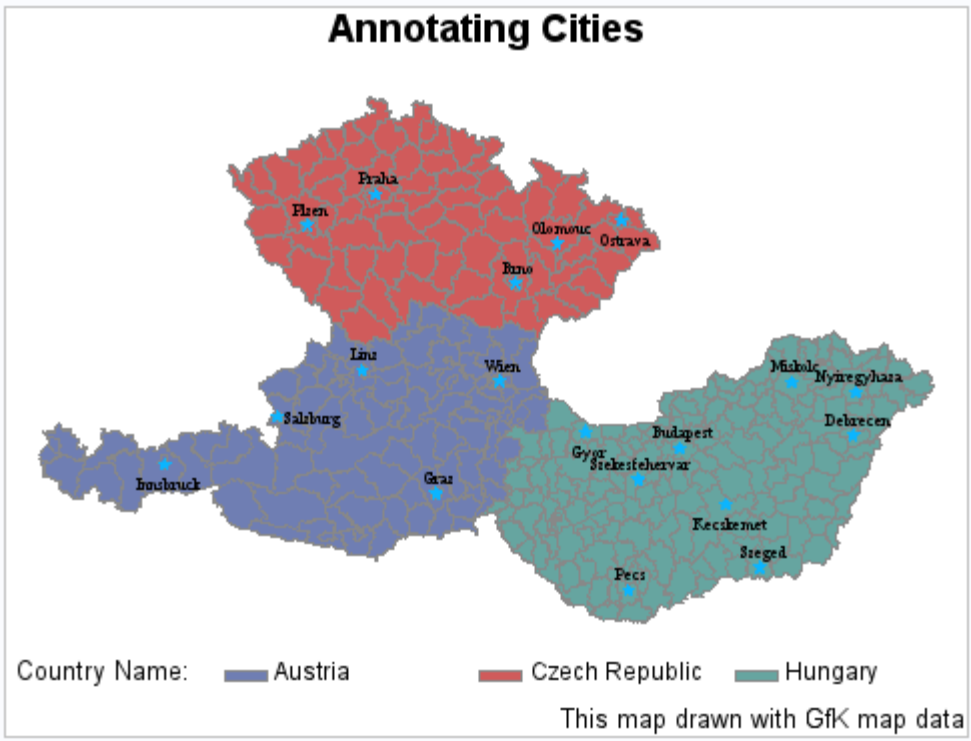
Restriction: The GfK GeoMarketing map data set used in this example is licensed to be used only with SAS/GRAPH.

Sample library member: GMPGCITA

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output



Program

```
data all;
```

```
length name $18;
set mapsgfk.austria(in=k1) mapsgfk.czech_republic(in=k2) mapsgfk.hungary;

if k1 then name='Austria';
else if k2 then name='Czech Republic';
else name='Hungary';
x=long;
y=lat;
run;

proc gproject data=all out=map degrees eastlong parmout;
id name id;
run;

data cities;
set mapsgfk.world_cities(where=(rank=1 and isoname in:('Austria' 'Czech' 'Hung')));
x=long;
y=lat;
run;

data labels;
   length color $8 text $55;
   retain style 'Thorndale AMT' function 'label' xsys ysys '2' hsys '3' when 'a';
   set cities;
   text='V';
   style='marker';
   color='CX0BB5FF';
   size=2;
   position='5';
   output;
   text='  '||city;
   position='2';
   size=2.25;
   style=' ';
   color='black';
   style='Thorndale AMT';
   if city in:('Gyor' 'Ostrav' 'Inns' 'Kecs') then position='8';
   if city in:('Salz') then position='6';

   output;
run;

proc gproject parmentry=map data=labels out=outdset dupok;
id;
run;

goptions reset=all border;

title1 "Annotating Cities";
footnote j=r "This map drawn with GfK map data";
legend1 label=("Country Name:");

proc gmap data=map map=map;
   id id;
   choro name / discrete
                xsize=95 pct
                coutline=gray88
                annotate=outdset
                des=''
```

```
                 name='eursub'
                 stat=first
                 legend=legend1;
run;
quit;
```

Program Description

Specify the countries to map.

```
data all;
length name $18;
set mapsgfk.austria(in=k1) mapsgfk.czech_republic(in=k2) mapsgfk.hungary;

if k1 then name='Austria';
else if k2 then name='Czech Republic';
else name='Hungary';
x=long;
y=lat;
run;
```

Project the map and store the projection parameters.

```
proc gproject data=all out=map degrees eastlong parmout;
id name id;
run;
```

Create the CITIES data set specifying cities to label (Annotate).

```
data cities;
set mapsgfk.world_cities(where=(rank=1 and isoname in:('Austria' 'Czech' 'Hung')));
x=long;
y=lat;
run;
```

Add Annotate variables to the CITIES data set.

```
data labels;
   length color $8 text $55;
   retain style 'Thorndale AMT' function 'label' xsys ysys '2' hsys '3' when 'a';
   set cities;
   text='V';
   style='marker';
   color='CX0BB5FF';
   size=2;
   position='5';
   output;
   text='  '||city;
   position='2';
   size=2.25;
   style=' ';
   color='black';
   style='Thorndale AMT';
   if city in:('Gyor' 'Ostrav' 'Inns' 'Kecs') then position='8';
   if city in:('Salz') then position='6';

   output;
```

```
run;
```

Project the annotation data set using the projection parameters stored in the prior map projection.

```
proc gproject parmentry=map data=labels out=outdset dupok;
id;
run;
```

Set the graphics environment.

```
goptions reset=all border;
```

Define the title, footnote, and legend for the map. The LEGEND= statement relabels the NAME variable.

```
title1 "Annotating Cities";
footnote j=r "This map drawn with GfK map data";
legend1 label=("Country Name:");
```

Produce the choropleth map that annotates cities in 3 European countries. The DISCRETE= option generates a separate color for each different response variable value (country name). The XSIZE= option specifies the width of the map being drawn. The COUTLINE= option outlines the map area regions in each country. The ANNOTATE= specifies the OUTDSET data set to annotate onto the map produced by the CHORO statement. The DESCRIPTION= option specifies that there is to be no chart description for web output. The NAME= option specifies the name of the graphics output file. The STATISTIC= option specifies that the GMAP procedure is to match the first observation from the projected data set and display the response value from that observation only. The LEGEND= option pulls in the LEGEND statement's label assignment.

```
proc gmap data=map map=map;
   id id;
   choro name / discrete
                xsize=95 pct
                coutline=gray88
                annotate=outdset
                des=''
                name='eursub'
                stat=first
                legend=legend1;
run;
quit;
```

Example 25: Using GfK GeoMarketing Map Data to Produce a Choropleth County Map Annotating Roads and Water

Features: MAP= required argument referring to GfK map data

DATA= argument referring to response data

ANNOTATE= argument referring to Annotate data

ID statement

CHORO statement options

- DISCRETE
- NOLEGEND
- ANNOTATE=

Other features: Annotate Facility
MAPIMPORT procedure
FREQ procedure
PATTERN statement
NOTE statement
FOOTNOTE statement

Data sets: MAPSGFK.US_COUNTIES (map data)
MAP (response data)

Restriction: The GfK GeoMarketing map data set used in this example is licensed to be used only with SAS/GRAPH.

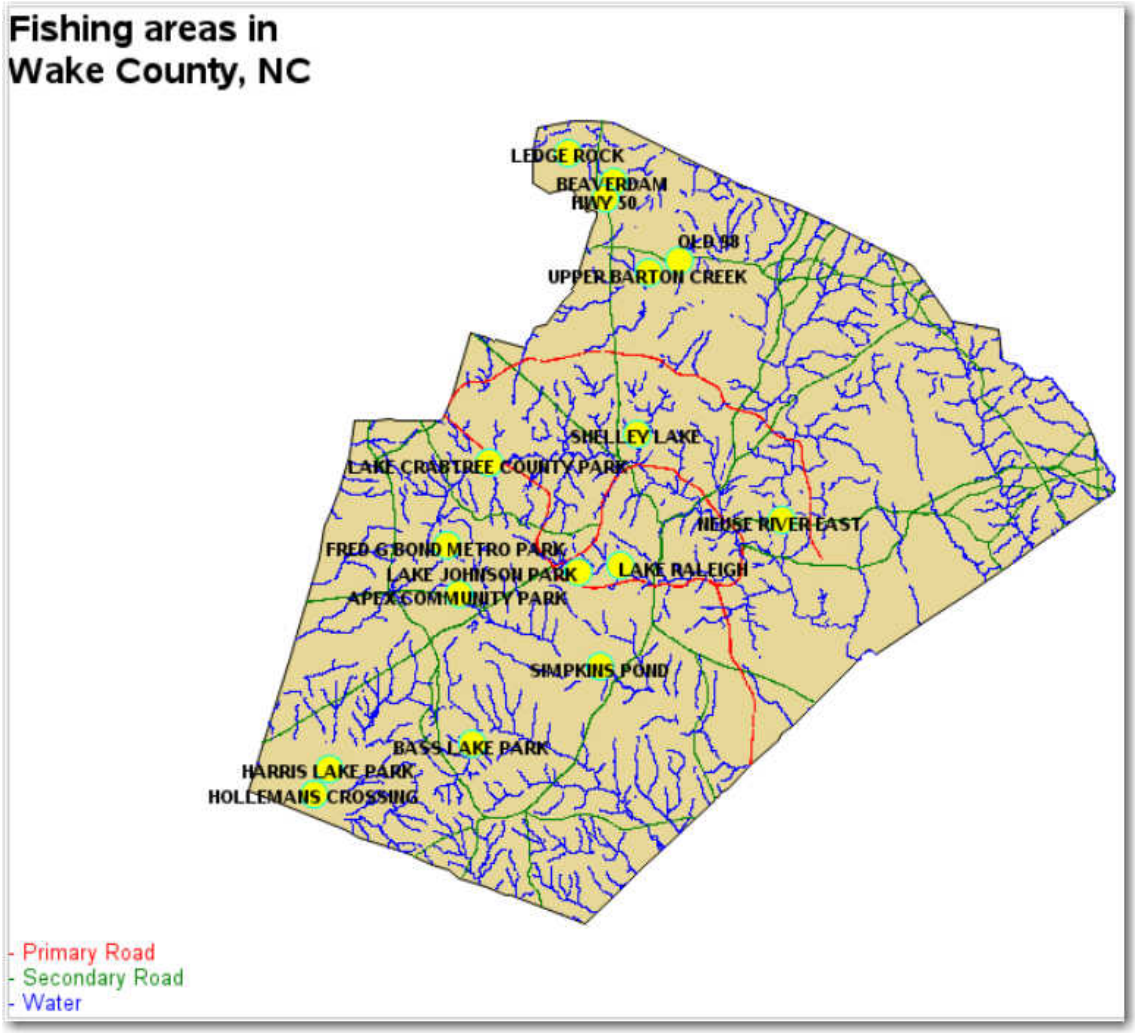
Sample library member: GMPGCHAN

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

Output

This example displays all the fishing areas in Wake County, in the state of North Carolina. A subset of a GfK map displays just Wake County, and then downloaded data displaying roads and water are annotated onto the map. You can download a zipped file of the TIGER/Line shapefile data from SAS Maps Online, located at the following URL:http://support.sas.com/rnd/datavisualization/mapsonline/html/sampledata.html. Look for the filename GMPGCHAN.ZIP. Or you can download the most recent TIGER/Line shapefile data from the U.S. Census Bureau website. The files used in the following example are named *tl_2012_37183_edges.xxx*, where *xxx* specifies the various shapefile extensions. See *SAS/GIS: Spatial Data and Procedure Guide* for more information about downloading and preparing the data. You must update the following code to specify the path where you stored the downloaded files before running this example.



Program

```
goptions reset=all border;

data ncwake;
length dsn $8;
  set mapsgfk.us_counties(where=(fipstate(state)="NC" and county=183));
  x=long;
  y=lat;
  dsn='map';
run;

proc mapimport datafile='path-to-shapefile'
out=work.wake;
run;

data roads;
  set work.wake(where=(mtfcc in:('S1100' 'S1200' 'H')));
run;

/*
proc ginside map=ncwake data=roads out=rdsinwake insideonly;
id id;
run;
*/
```

```
data annor(drop=statefp countyfp);
length STATE COUNTY 5 text color $8;
  retain xsys ysys '2' when 'A';

  /*set rdsinwake;*/ /* Uncomment this if you run the GINSIDE proc.*/

set roads; /* Comment out this data set if you run GINSIDE proc.*/
   dsn='lines';
  /* Define SAS variables: SAS data sets have */
  /* state and county defined as numeric     */
  STATE=statefp; county=countyfp;
  id='US-'||put(state,z2.)||put(county,z3.);

  LONG=X;  LAT=Y;

  by TLID notsorted;
  text=mtfcc;
  if mtfcc='S1100' then do; line=3; color='red'; size=1.25; end;
  else if mtfcc='S1200' then do; line=2; color='green';end;
  else if mtfcc=:'H' then do; line=4; color='blue'; end;

  if first.TLID then do;
   function='MOVE';
   output;
  end;
  else do;
   function='DRAW';
   output;
  end;
run;

data WORK.WRC_FISHING;
infile cards dsd missover;
input X Y Fishing_Area_Name : $25.
          FishingPier
          ShorelineAccess
          BoatRamp
          CanoeAccess
          UniveralAccess
          WaterbodyName & $29.;

cards;
-78.5392, 35.7987,NEUSE RIVER EAST,0, 1, 0, 1, 0, NEUSE RIVER
-78.6275, 35.9791,OLD 98,0, 1, 0, 0, 0, FALLS RESERVOIR
-78.6540, 35.9705,UPPER BARTON CREEK,0, 1, 0, 1, 0, FALLS RESERVOIR
-78.7226, 36.0536,LEDGE ROCK,., 0, 1, 0, 0, FALLS RESERVOIR
-78.6843, 36.0342,BEAVERDAM,0, 1, 1, 1, 0, FALLS RESERVOIR
-78.6915, 36.0215,HWY 50,0, 1, 1, 1, 0, FALLS RESERVOIR
-78.8166, 35.7468,APEX COMMUNITY PARK,1, 1, 0, 1, 0, PINE LAKE
-78.8042, 35.6429,BASS LAKE PARK,1, 1, 0, 1, 0, BASS LAKE PARK
-78.9390, 35.6086,HOLLEMANS CROSSING,., 0, 1, 0, 0, SHEARON
HARRIS RESERVOIR
-78.6957, 35.6972,SIMPKINS POND,0, 1, 0, 0, 0, SIMPKINS POND
/ LAKE WHEELER
-78.6635, 35.8585,SHELLEY LAKE,1, 1, 1, 1, 1, SHELLEY LAKE
-78.6775, 35.7668,LAKE RALEIGH,1, 1, 0, 1, 1, LAKE RALEIGH
```

```
-78.7907, 35.8384,LAKE CRABTREE COUNTY PARK,1, 1, 1, 1, 1, LAKE
CRABREE COUNTY PARK
-78.7137, 35.7627,LAKE JOHNSON PARK,1, 0, 0, 1, 0, LAKE JOHNSON
-78.9259, 35.6258,HARRIS LAKE PARK,1, 1, 0, 1, 0, HARRIS LAKE
& HARRIS PARK POND
-78.8259, 35.7812,FRED G BOND METRO PARK,1, 1, 0, 1, 0, BOND
PARK POND
;
run;

data wrc;
length STATE 5 COUNTY 5 text $45 font $8 color $8
       function $8 dsn $8;
retain STATE 37 COUNTY 183 xsys ysys '2' when 'A'
       font 'marker';
set wrc_fishing;
;

 dsn='points';

text=fishing_area_name;
style='Albany AMT/bold';
function='label';
 /* reposition overlapping points */
if text in('LAKE RALEIGH') then position='6';
else if text in:('LAKE JOHNSON') then position='4';
else if text=:'OLD 98' then position='3';
output;
function='pie';
rotate=360;
style='psolid';
color='yellow';
position='5';
size=.75;
output;
  /* Draw a dark ring around pie, to help */
  /* distinguish overlapping ones         */
style='pempty';
color='cyan';
line=1;
html='';
output;

run;

data all;
length function $8 text $45 style $20;
set ncwake annor wrc;
run;

proc gproject data=all out=prj degrees eastlong dupok;
id dsn id;
run;

data map lines points;
set prj;
if dsn='map' then output map;
```

```
else if dsn='lines' then output lines;
else output points;
run;

proc gmap map=map data=map anno=lines;
   id id; choro state/discrete nolegend anno=points stat=sum;
   pattern v=s r=100 color=cream;
   title1 j=left font='Albany AMT/bold' height=2 'Fishing areas in';
   title2 j=left font='Albany AMT/bold' height=2 'Wake County, NC';
   footnote j=l c=red '- Primary Road';
   footnote2 j=l c=green '- Secondary Road';
   footnote3 j=l c=blue '- Water';
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Subset Wake County in the state of North Carolina from the GfK map data.

```
data ncwake;
length dsn $8;
  set mapsgfk.us_counties(where=(fipstate(state)="NC" and county=183));
  x=long;
  y=lat;
  dsn='map';
run;
```

Use the MAPIMPORT procedure to produce an output map data set from data that you downloaded from the U.S. Census Bureau TIGER website. Specify the path and filename of the downloaded shapefiles. An example shapefile name is "tl_2012_37183_edges". The data includes roads and water.

```
proc mapimport datafile='path-to-shapefile'
out=work.wake;
run;
```

Keep the primary and secondary roads and the water lines from the downloaded shapefile.

```
data roads;
  set work.wake(where=(mtfcc in:('S1100' 'S1200' 'H')));
run;
```

Optional step. Running the GINSIDE procedure is necessary only if the Annotate step that is run next results in "observation mapping errors". In order to run GINSIDE, you must first remove the comments. Run this procedure to keep those road points that are only within Wake County map boundaries.

```
/*
proc ginside map=ncwake data=roads out=rdsinwake insideonly;
id id;
run;
*/
```

Create an Annotate data set so that the roads and fishing areas can be displayed on top of the map. Note: If you chose to run the prior GINSIDE procedure, then switch the data set that is designated. See the comment text in the code for directions. TLID is the edge ID, which can be used to connect to other TIGER files.

```
data annor(drop=statefp countyfp);
length STATE COUNTY 5 text color $8;
  retain xsys ysys '2' when 'A';

  /*set rdsinwake;*/ /* Uncomment this if you run the GINSIDE proc.*/

set roads; /* Comment out this data set if you run GINSIDE proc.*/
   dsn='lines';
  /* Define SAS variables: SAS data sets have */
  /* state and county defined as numeric     */
  STATE=statefp; county=countyfp;
  id='US-'||put(state,z2.)||put(county,z3.);

  LONG=X;  LAT=Y;

  by TLID notsorted;
  text=mtfcc;
  if mtfcc='S1100' then do; line=3; color='red'; size=1.25; end;
  else if mtfcc='S1200' then do; line=2; color='green';end;
  else if mtfcc=:'H' then do; line=4; color='blue'; end;

  if first.TLID then do;
   function='MOVE';
   output;
  end;
  else do;
   function='DRAW';
   output;
  end;
run;
```

Create the data set WORK.WRC_FISHING that contains the NC fishing areas. The source is the North Carolina Wildlife Resource Center.

```
data WORK.WRC_FISHING;
infile cards dsd missover;
input X Y Fishing_Area_Name : $25.
          FishingPier
          ShorelineAccess
          BoatRamp
          CanoeAccess
          UniveralAccess
          WaterbodyName & $29.;

cards;
-78.5392, 35.7987,NEUSE RIVER EAST,0, 1, 0, 1, 0, NEUSE RIVER
-78.6275, 35.9791,OLD 98,0, 1, 0, 0, 0, FALLS RESERVOIR
-78.6540, 35.9705,UPPER BARTON CREEK,0, 1, 0, 1, 0, FALLS RESERVOIR
-78.7226, 36.0536,LEDGE ROCK,., 0, 1, 0, 0, FALLS RESERVOIR
-78.6843, 36.0342,BEAVERDAM,0, 1, 1, 1, 0, FALLS RESERVOIR
-78.6915, 36.0215,HWY 50,0, 1, 1, 1, 0, FALLS RESERVOIR
```

```
-78.8166, 35.7468,APEX COMMUNITY PARK,1, 1, 0, 1, 0, PINE LAKE
-78.8042, 35.6429,BASS LAKE PARK,1, 1, 0, 1, 0, BASS LAKE PARK
-78.9390, 35.6086,HOLLEMANS CROSSING,., 0, 1, 0, 0, SHEARON
HARRIS RESERVOIR
-78.6957, 35.6972,SIMPKINS POND,0, 1, 0, 0, 0, SIMPKINS POND
/ LAKE WHEELER
-78.6635, 35.8585,SHELLEY LAKE,1, 1, 1, 1, 1, SHELLEY LAKE
-78.6775, 35.7668,LAKE RALEIGH,1, 1, 0, 1, 1, LAKE RALEIGH
-78.7907, 35.8384,LAKE CRABTREE COUNTY PARK,1, 1, 1, 1, 1, LAKE
CRABREE COUNTY PARK
-78.7137, 35.7627,LAKE JOHNSON PARK,1, 0, 0, 1, 0, LAKE JOHNSON
-78.9259, 35.6258,HARRIS LAKE PARK,1, 1, 0, 1, 0, HARRIS LAKE
& HARRIS PARK POND
-78.8259, 35.7812,FRED G BOND METRO PARK,1, 1, 0, 1, 0, BOND
PARK POND
;
run;
```

Add the Annotate variables.

```
data wrc;
length STATE 5 COUNTY 5 text $45 font $8 color $8
       function $8 dsn $8;
retain STATE 37 COUNTY 183 xsys ysys '2' when 'A'
       font 'marker';
set wrc_fishing;
;

 dsn='points';

text=fishing_area_name;
style='Albany AMT/bold';
function='label';
 /* reposition overlapping points */
if text in('LAKE RALEIGH') then position='6';
else if text in:('LAKE JOHNSON') then position='4';
else if text=:'OLD 98' then position='3';
output;
function='pie';
rotate=360;
style='psolid';
color='yellow';
position='5';
size=.75;
output;
  /* Draw a dark ring around pie, to help */
  /* distinguish overlapping ones         */
style='pempty';
color='cyan';
line=1;
html='';
output;

run;
```

Combine the map and roads data sets so that they can be projected together.

```
data all;
length function $8 text $45 style $20;
set ncwake annor wrc;
run;

proc gproject data=all out=prj degrees eastlong dupok;
id dsn id;
run;
```

Separate the map and Annotate data sets.

```
data map lines points;
set prj;
if dsn='map' then output map;
else if dsn='lines' then output lines;
else output points;
run;
```

Display the choropleth map.

```
proc gmap map=map data=map anno=lines;
   id id; choro state/discrete nolegend anno=points stat=sum;
   pattern v=s r=100 color=cream;
   title1 j=left font='Albany AMT/bold' height=2 'Fishing areas in';
   title2 j=left font='Albany AMT/bold' height=2 'Wake County, NC';
   footnote j=l c=red '- Primary Road';
   footnote2 j=l c=green '- Secondary Road';
   footnote3 j=l c=blue '- Water';
run;
quit;
```

Example 26: Labeling Cities on a Map

Features: Annotate functions
- LABEL
- SYMBOL

Annotate variables
- HSYS
- POSITION
- SIZE
- TEXT
- WHEN
- X and Y
- XSYS
- YSYS

Data set: MAPSGFK.USCITY

Sample library member: GANCITY

Note: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

This example labels a map of the continental United States with the location and names of three cities. The GMAP procedure draws a map of the U.S. and an Annotate data set adds the stars and labels.

The DATA step that creates the Annotate data set gets the *x* and *y* coordinates of the cities to be labeled from the MAPSGFK.USCITY data set. Because MAPSGFK.USCITY stores projected coordinates in the X and Y variables, the DATA step does not need to reassign the variable values. Also, because X and Y contain data values (the map data set coordinates), the XSYS and YSYS variables specify coordinate system 2, absolute data values. However, the HSYS variable that controls text height uses coordinate system 3, percent of the graphics output area.

See "Example 4: Projecting an Annotate Data Set" on page 384 for an example of labeling a map using map coordinates in units of latitude and longitude.

See "Differences between GfK and Traditional Map Data Sets " on page 195for more information about using map data sets.

Output 7.1 *Map with Labeled Cities*

Program

```
goptions reset=all border;

data lower48;
   set maps.us;
   if state ne stfips("AK");
   if state ne stfips("HI");
   if state ne stfips("PR");
run;

data citystar;
   length function style color $ 8 position $ 1
          text $ 20;
   retain xsys ysys "2" hsys "3"
          when "a";

   set mapsgfk.uscity(keep=x y city state);
```

```
   if (city="Atlanta" and state=13)
       or (city="Chicago" and state=17)
       or (city="Seattle" and state=53);

   function="symbol"; style="marker"; text="V"; color="red"; size=5;
      output;

   function="label"; style=""; text=city; color="green";
      size=5; position="8"; output;
run;

title "Distribution Center Locations";

pattern value=mempty color=blue repeat=49;

proc gmap data=lower48 map=lower48;
   id state;
   choro state / annotate=citystar discrete nolegend;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Subset the U.S. map data set by omitting Alaska, Hawaii, and Puerto Rico.

```
data lower48;
   set maps.us;
   if state ne stfips("AK");
   if state ne stfips("HI");
   if state ne stfips("PR");
run;
```

Create the Annotate data set, CITYSTAR. CITYSTAR contains the commands that draw a star and a label at each of the three cities. Setting WHEN to A draws the annotation after the map.

```
data citystar;
   length function style color $ 8 position $ 1
          text $ 20;
   retain xsys ysys "2" hsys "3"
          when "a";
```

Include the values of selected variables from MAPSGFK.USCITY. X and Y contain projected coordinates; CITY contains names; STATE contains FIPS codes. Because there are several instances of Atlanta, a STATE value is necessary. To avoid multiple symbols being displayed for the same city name in different states, a STATE value specifies the target state.

```
   set mapsgfk.uscity(keep=x y city state);
   if (city="Atlanta" and state=13)
       or (city="Chicago" and state=17)
       or (city="Seattle" and state=53);
```

Create the observation that draws the star. The text string V is the character code for the star figure in the MARKER font assigned by the STYLE variable.

```
      function="symbol"; style="marker"; text="V"; color="red"; size=5;
         output;
```

Create the observation that labels the city. TEXT is assigned the value of CITY. The default font is used. SIZE uses the units assigned by HSYS, so text height is 5% of the height of the graphics output area. POSITION 8 places the label directly below the city location.

```
      function="label"; style=""; text=city; color="green";
         size=5; position="8"; output;
   run;
```

Define the title for the map.

```
   title "Distribution Center Locations";
```

Define patterns for the map areas. MEMPTY colors only the state borders.

```
   pattern value=mempty color=blue repeat=49;
```

Generate the map and assign the annotate data set to the CHORO statement.

```
   proc gmap data=lower48 map=lower48;
      id state;
      choro state / annotate=citystar discrete nolegend;
   run;
   quit;
```

Example 27: Producing an SVG File That Displays a European Map

Features: MAP= required argument referring to MAPSSAS map data

DATA= argument referring to response data

ANNOTATE= argument referring to Annotate data

ID statement

WHERE= argument

CHORO statement options
- NAME=

Other features
- TITLE statement
- ODS HTML statement
- GOPTIONS statement with DEVICE= option

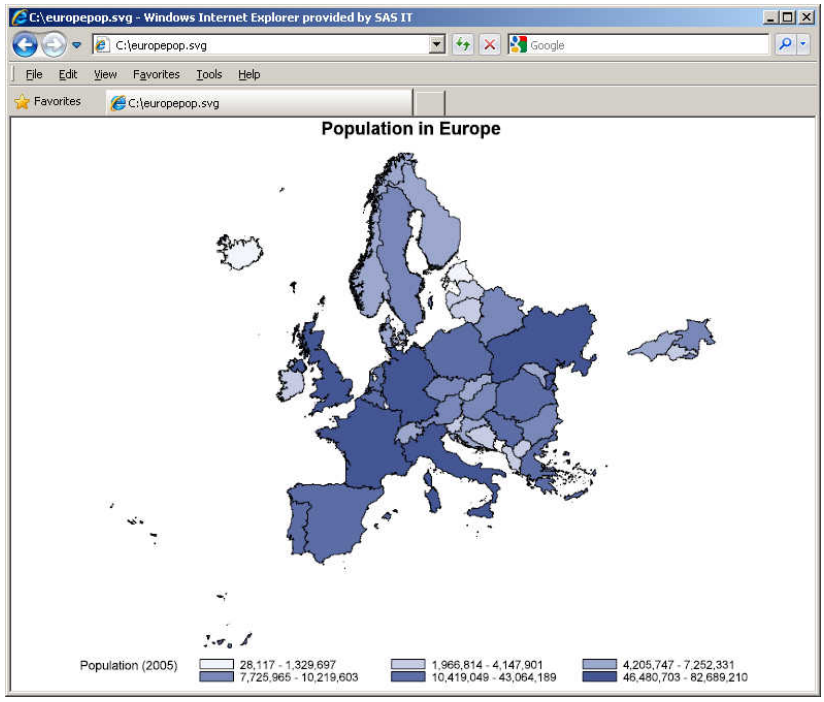
Data set: MAPSSAS

Output

This example produces an SVG file named **europepop.svg** and an HTML4 file named **europe.htm**.

You can view the SVG graphic by opening **europe.htm** in your SVG-enabled browser. If your browser does not render the graphic, see "Browser Support for Viewing SVG Files" in *SAS/GRAPH: Reference*.

You can view the SVG coding by opening the SVG file, **europepop.svg**, in a text editor.



Specify the ODS output path. The SVG file is written to the path designated by your installation.

```
filename odsout ".";
```

Set the graphics environment and set the output device to SVG.

```
goptions reset=all device=svg;
```

Set the output destination to HTML4, specify the path to file the output, and specify a name for the output file. The output filename is “europe.htm”.

```
ods _all_ close;
ods html4 path=odsout file="europe.htm";
```

Set a title to display on the graphic output.

```
title "Population in Europe";
```

Use the GMAP procedure to produce a choropleth map from the MAPSSAS European map data set, using population statistics from the SASHELP.DEMOGRAPHICS data set. The map is to exclude certain countries like Greenland.

```
proc gmap map=mapssas.europe(where=(id ne 405 and id ne 845))
                 data=sashelp.demographics(where=(cont=93)) all;
    id id;
    choro pop / name="europePop";
run;
quit;
ods html4 close;
ods html; /* Not required in SAS Studio */
```

Example 28: Producing a Stand-Alone SVG File in HTML5 That Displays a European Map

Features: MAP= required argument referring to MAPSSAS map data

DATA= argument referring to response data

ANNOTATE= argument referring to Annotate data

ID statement

WHERE= argument

CHORO statement options
- NAME=

Other features
- TITLE statement
- ODS HTML statement
- SVG_MODE=embed option
- GOPTIONS statement with DEVICE= option

Data set: MAPSSAS

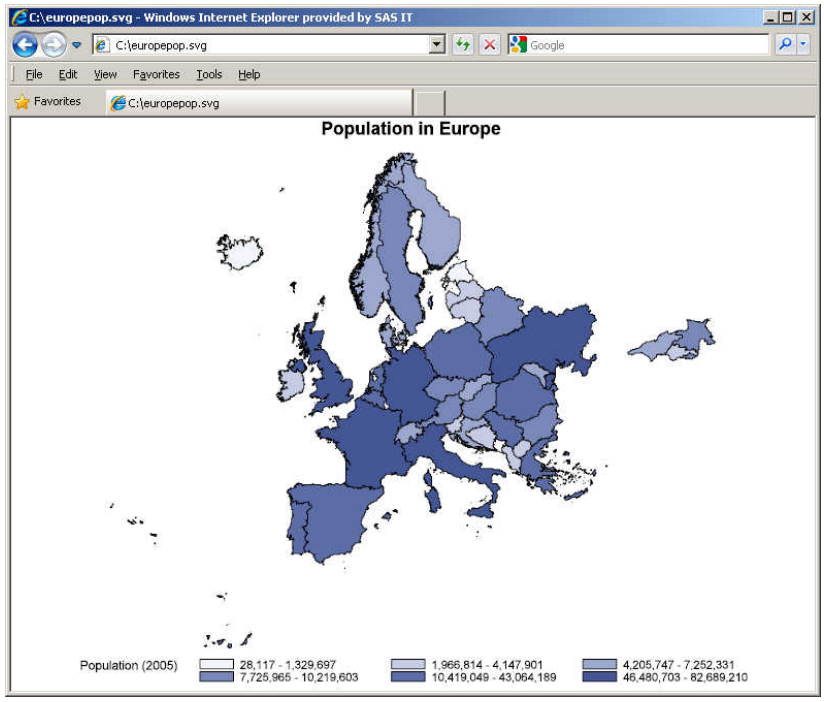
Output

In the HTML4 destination, SVG graphics are created as stand-alone files by default. In the HTML5 destination, SVG graphics are created as inline graphics in the HTML file. Starting with the fourth maintenance release for SAS 9.4, the ODS HTML statement refers to an HTML5 destination by default. You can generate a stand-alone SVG image file in the ODS HTML destination by specifying **`svg_mode='embed'`** in the ODS HTML statement. This option embeds the SVG image file into the HTML file using the **`<embed>`** tag. In the following example, the code shown in "Example 27: Producing an SVG File That Displays a European Map" on page 350 is modified to use this option.

This example produces a stand-alone SVG file named **`europepop.svg`** and an HTML5 file named **`europe.htm`**.

You can view the SVG graphic by opening **`europe.htm`** in your SVG-enabled browser. If your browser does not render the graphic, see "Browser Support for Viewing SVG Files" in *SAS/GRAPH: Reference*.

You can view the SVG coding by opening the SVG file, **`europepop.svg`**, in a text editor.



Specify the ODS output path. The SVG file is written to the path designated by your installation.

```
filename odsout ".";
```

Set the graphics environment and set the output device to SVG.

```
goptions reset=all device=svg;
```

Set the output destination to HTML5, specify the path to which the file is written, and specify a name for the output file. The output filename is “europe.htm”.

```
ods _all_ close;
ods html options(svg_mode='embed') path=odsout file="europe.htm";
```

Set a title to display on the graphic output.

```
title "Population in Europe";
```

Use the GMAP procedure to produce a choropleth map from the MAPSSAS European map data set, using population statistics from the SASHELP.DEMOGRAPHICS data set. The map is to exclude certain countries like Greenland.

```
proc gmap map=mapssas.europe(where=(id ne 405 and id ne 845))
                 data=sashelp.demographics(where=(cont=93)) all;
    id id;
    choro pop / name="europePop";
run;
quit;
ods html close;
ods html; /* Not required in SAS Studio */
```

Chapter 8

GPROJECT Procedure

Overview: GPROJECT Procedure

The GPROJECT procedure processes map data sets by converting spherical coordinates (longitude and latitude) into Cartesian coordinates for use by the GMAP procedure. The process of converting coordinates from spherical to Cartesian is called projecting. All of the GfK and many of the traditional map data sets that are available with SAS/GRAPH

contain unprojected longitude and latitude coordinates. When these coordinates are plotted by the GMAP procedure, which is designed to plot points on a two-dimensional plane, the resulting map is often reversed and distorted as a result of forcing the spherical map coordinates onto a flat plane.

The GPROJECT procedure enables you to use one of several map projection techniques to project the latitude and longitude coordinates onto a two-dimensional plane. It does this while attempting to minimize the distortion of area, distance, direction, and shape properties of the original sphere. The output map data set that is produced by the GPROJECT procedure contains Cartesian coordinates that can be displayed correctly using the GMAP procedure.

The GPROJECT procedure can also create a rectangular subset of the input map data set. It does this by excluding all points with longitude and latitude values that fall outside of a specified range. This provides a simple way to reduce the size of the map data set if you need only a portion of a larger map.

The GPROJECT procedure does not produce any graphics output. Instead, it produces an output map data set, which can be used as the input map data set for the GMAP procedure. See Chapter 7, "GMAP Procedure," on page 189.

Figure 8.1 on page 356 and Figure 8.2 on page 357 illustrate the effect of using GPROJECT defaults (Albers projection with standard parallels that are calculated by the procedure) to project a typical map data set with coordinates that are stored as longitude and latitude.

The program for the following maps can be seen in "Example 1: Using Default Projection Specifications" on page 376.

Figure 8.1 *Map before Projection (GPJDEFLT(a))*

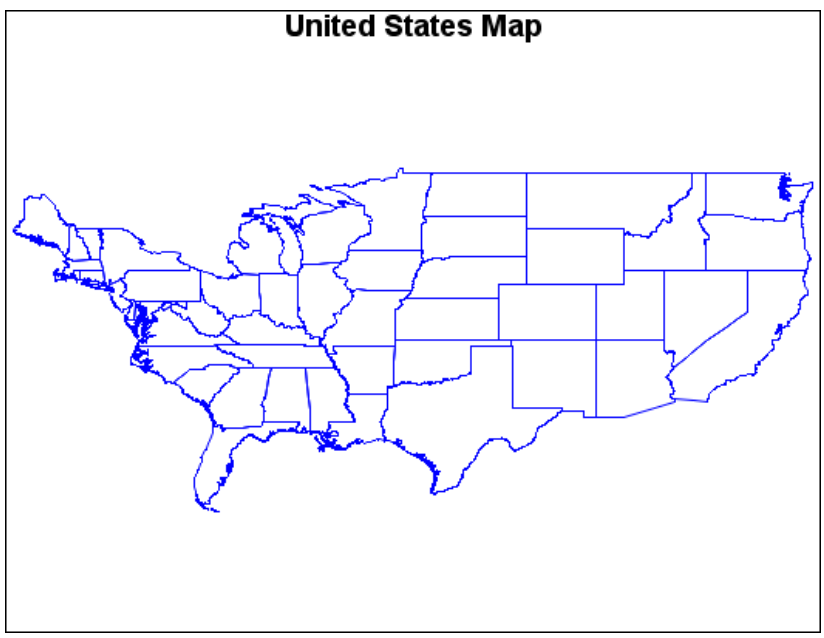
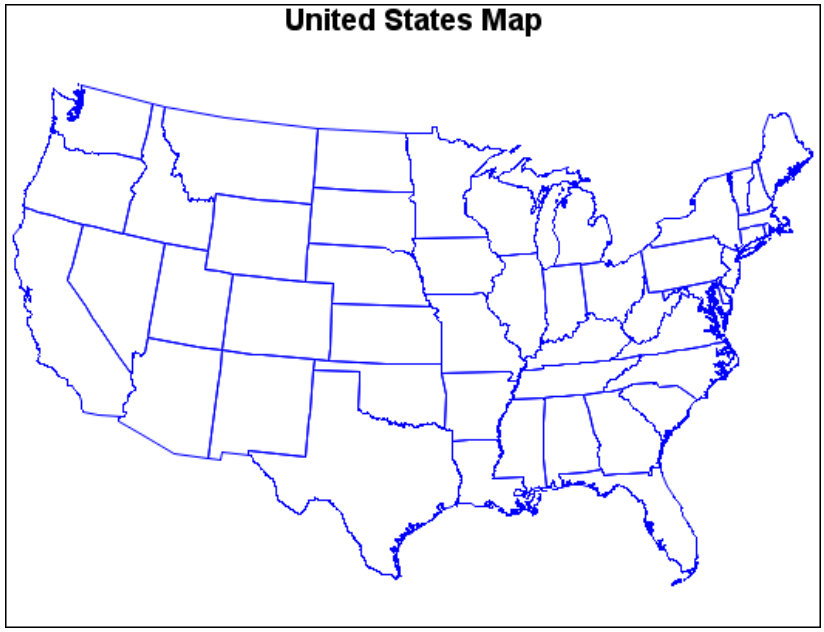


Figure 8.2 *Map after Projection (GPJDEFLT(b))*



About the Input Map Data Set

Input Map Data Set Variables

The input map data set must be in a SAS data set format (see "About Map Data Sets" on page 193), and it must contain these variables:

- a numeric variable named X that contains the longitude coordinates of the map boundary points.
- a numeric variable named Y that contains the latitude coordinates of the map boundary points.
- one or more identification variables that uniquely identify the unit areas in the map. These variables are listed in the ID statement.

The X and Y variables contain the values that are to be projected.

In addition, the input map data set can also contain these variables:

- a numeric variable named SEGMENT that distinguishes nonconterminous segments of the unit areas.
- a numeric variable named DENSITY that can be used to affect the output from PROC GPROJECT. See "Clipping Map Data Sets" on page 375 for more information.

Other variables in the input map data set do not affect the GPROJECT procedure.

Input Map Data Sets That Contain Only Unprojected Values

The following is a list of all of the data sets supplied by SAS that contain X and Y variables whose values are unprojected:

- CANADA3
- CANADA4
- COUNTIES
- COUNTY
- STATES

See "Example 1: Using Default Projection Specifications" on page 376 for an illustration of this type of input map data set and the variables that it contains.

Note: Projection is appropriate for map data sets in which the X and Y variable values represent longitude and latitude. Some of the map data sets that are supplied with SAS/GRAPH have already been projected; such data set should not be projected again.

Input Map Data Sets That Contain Both Projected and Unprojected Values

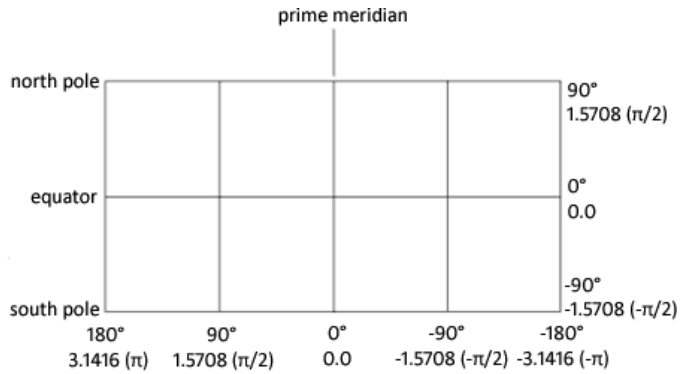
All GfK GeoMarketing and most traditional map data sets contain both sets of variables (X, Y, and LONG, LAT) for projected and unprojected maps. In these cases, the X and Y variables produce a projected map, so you do not need to use the GPROJECT procedure. However, you might want to use the LONG and LAT variables to reproject the map using a different projection type. The LATLON option specifies that the GPROJECT procedure uses the LONG and LAT variables instead of X and Y.

For additional information about the supplied SAS/GRAPH map data sets, see "About Map Data Sets" on page 193 and the METAMAPS data set in your maps data set directory.

About Coordinate Values

Figure 8.3 on page 359 shows the standard coordinate system for map data sets with coordinates in longitude and latitude. For the longitude and latitude values (below and to the right of the figure, respectively) the upper value is expressed in degrees and the lower value is expressed in radians. A radian is approximately 57.3 degrees.

Figure 8.3 *Longitude and Latitude Coordinates*



By default, the GPROJECT procedure assumes that the units for the input coordinate values are radians. It also assumes that values for the horizontal coordinate increase from east to west across the map. If your map coordinates are stored as degrees of arc, use the DEGREE option in the PROC GPROJECT statement. If the horizontal coordinate values in the map increase west to east rather than east to west, use the EASTLONG option in the PROC GPROJECT statement.

The traditional unprojected map data sets that are provided with SAS/GRAPH can be projected if you use the default procedure characteristics: coordinate units in the data sets are radians, and horizontal values increase east to west. The MAPSGFK map data sets that are provided with SAS/GRAPH use degrees as the coordinate units, and horizontal values (longitude) increase from west to east.

About Types of Map Projections

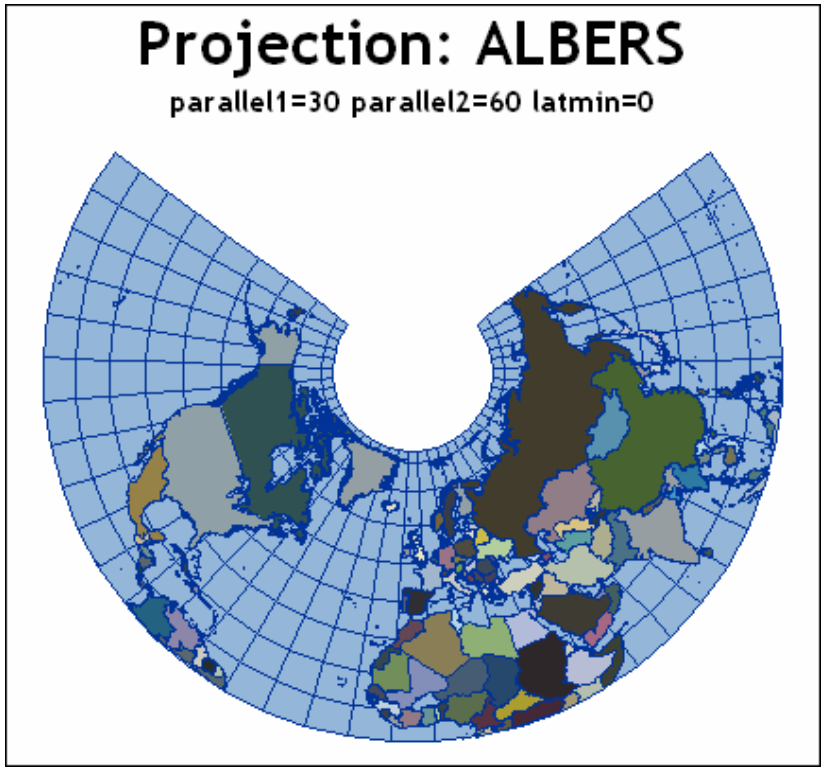
Projection Types

The GPROJECT procedure performs four different types of projection:

- Albers' equal-area projection with two standard parallels (the default method)
- Lambert's conformal projection with two standard parallels
- gnomonic projection (an azimuthal equidistant projection)
- proj.4 projection enables the transformation of geographic coordinates from one projection or datum to another projection or datum.

Albers' Equal-Area Projection

Figure 8.4 *Albers' Projection*



The Albers' projection is a conic projection from the surface of the sphere to a cone secant to the sphere, cutting it at two standard parallels of latitude. The axis of the cone coincides with an extension of the polar axis of the sphere. Each section of the resulting map bears a constant ratio to the area of the sphere. In general, distortion in shape tends to increase toward the poles in latitudes outside of the two standard parallels.

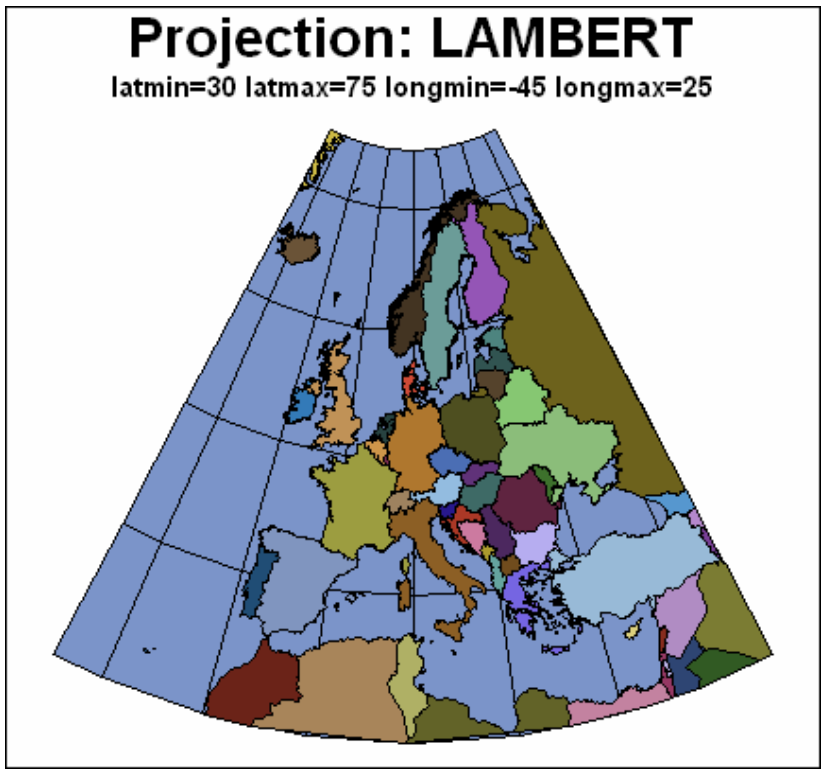
Figure 8.4 on page 360 illustrates an Albers' equal-area projection of the northern hemisphere.[1]

The Albers' projection is suitable for portraying large and small areas that extend east to west and produces satisfactory results in most cases. However, both standard parallels must lie on the same side of the equator, so this method might not be suitable for map data sets that extend north to south and span the equator. For those map data sets, use the gnomonic projection method.

Lambert's Conformal Projection

1 The projection examples in this topic include grid lines that were added with the Annotate facility. See the Samples area at support.sas.com for an example of adding latitude and longitude lines to a map.

Figure 8.5 *Lambert's Projection*



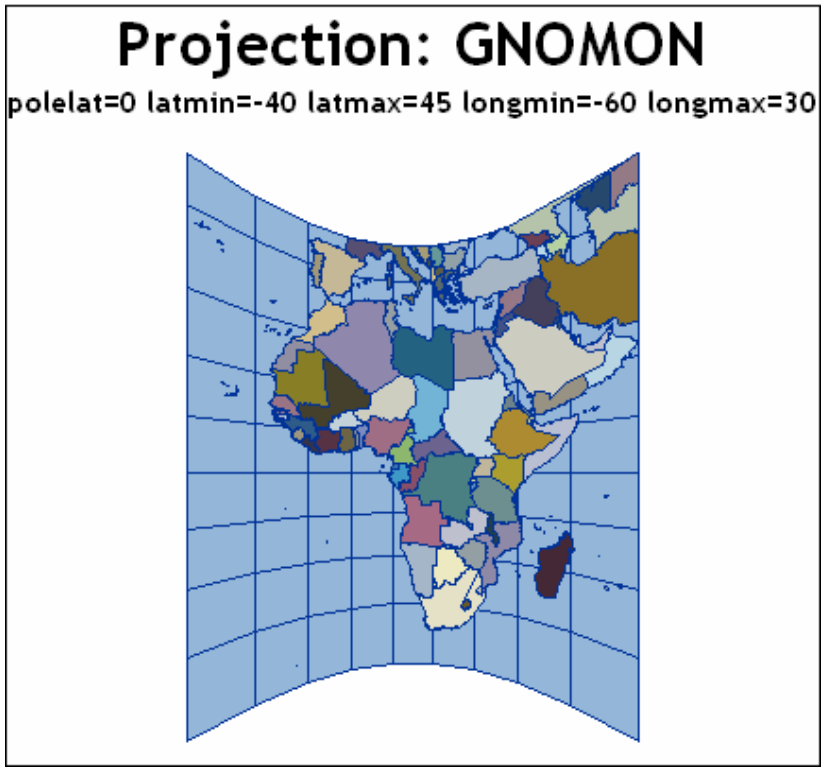
The Lambert's projection is obtained from a secant cone in the same manner as Albers' projection. In the Lambert's projection, meridians of longitude are straight lines that radiate from the apex of the cone, whereas parallels of latitude are concentric circles. The Lambert's projection is somewhat better than the Albers' projection at representing the original shape of projected unit areas. The Albers' projection is somewhat better at representing relative sizes of projected unit areas.

Figure 8.5 on page 361 illustrates a Lambert's conformal projection of Europe.

The Lambert's projection is ideal for navigational charts and relatively small maps that extend east to west. However, as in the Albers' projection, both standard parallels must lie on the same side of the equator. As a result, this method might not be suitable for map data sets that span the equator. For those map data sets, use the gnomonic projection method.

Gnomonic Projection

Figure 8.6 *Gnomonic Projection*



The gnomonic projection is a planar projection from the surface of the sphere directly onto an imaginary plane tangent to the sphere at the map projection pole. By default, the projection pole is placed at the center of the map data set that is to be projected. However, you can specify the projection pole to be anywhere on the surface of the sphere. (See the options "POLELAT=*latitude*POLELONG=*longitude*" on page 371.)

Figure 8.6 on page 362 illustrates a gnomonic projection of Africa.

In the gnomonic projection, distortion increases as the distance from the map pole increases. Because of this distortion, the PROC GPROJECT procedure deletes all of the observations that lie more than 85 degrees from the map pole. The gnomonic projection is most appropriate for mapping small areas that extend east to west.

Proj.4 Enables Conversion of Geographic Coordinates From One Projection or Datum to Another

The GPROJECT procedure can perform projections between any number of different projection types using the proj.4 system of projection strings. To do this specify PROJ4 on the PROJECT= option for the PROC GPROJECT statement. The proj.4 projection allows the transformation of geographic coordinates from one projection or datum to another projection or datum.

Proj.4 LAT and LONG to OSM (aka Mercator or UTM) Projection, and Vice Versa

The proj.4 projection, by default, enables a transformation from latitude and longitude geographic coordinates (EPSG:4326) to an OpenStreetMap (OSM) coordinate system. The OSM coordinate system is also known as the 900913, or Universal Transverse Mercator (UTM) coordinate system. Use the FROM or TO options on PROC

GPROJECT to override either of these defaults. For example, if you have a Mercator projection, you can use these options to revert the projection to EPSG:4326. For more information see the options "FROM="*coordinate-system*"" on page 367 and "TO="*coordinate-system*"" on page 372.

The EPSG:4326, or LAT and LONG, is a common latitude and longitude coordinate reference system that is also referred to as the World Geodetic System 1984 (WGS84). The 900913, or Web Mercator, is a cylindrical projection that maps meridians and parallels straight and perpendicular. Google and OSM use Web Mercator in their maps system.

TIP Make sure that all options begin with a plus sign "+".

Proj.4 Convert State Plane Coordinates to LAT and LONG Geographic Coordinates

Use the PROJ4 option of the GPROJECT procedure to convert state plane coordinates to latitude and longitude geographic coordinates (EPGS:4326), or World Geodetic System 1984 (WGS84). The map data sets provided by SAS utilize LAT and LONG coordinates. For example, convert North Carolina state plane coordinates to WSG84 latitude and longitude coordinates with the following code:

```
proc gproject
    from="+proj=lcc +lat_1=34.33333333333334 +lat_2=36.16666666666666
    +lat_0=33.75 +lon_0=-79 +x_0=609601.22 +y_0=0 +ellps=GRS80 +datum=NAD83
    +units=m" to="WGS84"
;
```

Search for "Proj.4" at http://spatialreference.org to find other examples of state plane proj.4 strings.

About Projection Parameters Data Sets

The PARMOUT= and PARMIN= options enable you to store and retrieve projection parameters from a data set. By using projection parameters, you can project two data sets in exactly the same way (for example, to project annotation points and a map separately).

Parameters data sets contain the following variables:

PROJ_MAP	identifies the map data set for the current observation.
PROJ_NAME	specifies the projection method
PROJ_OPTS	contains a space-delimited list of options.
PROJ_P1	specifies the value for the PARALLEL1 option.
PROJ_P2	specifies the value for the PARALLEL2 option.
PROJ_STDLAT	specifies the standard latitude for the equirectangular projection method.
PROJ_STDLONG	specifies the central meridian or standard longitude for the projection.
PROJ_XMIN	specifies the minimum X value for the projection.
PROJ_YMIN	specifies the minimum Y value for the projection.

Note: If any of the variable values are a missing value, then the values are calculated by the GPROJECT procedure.

Note: You can override the values of input projection parameters by specifying options in the PROC GPROJECT statement.

By default, the GPROJECT procedure automatically uses parameters from any data set named PROJPARM that is in the same library as the input data set. To disable input parameters, specify NOPARMIN.

Syntax: GPROJECT Procedure

Requirement: Exactly one ID statement is required.

PROC GPROJECT DATA=*input-map-data-set* <*option(s)*>;
ID *id-variable(s)*;

PROC GPROJECT Statement

Identifies the input and output map data sets. Can specify the type of projection, and the criteria for clipping and projection.

Requirement: An input map data set is required.

Syntax

PROC GPROJECT DATA=*input-map-data-set* <*option(s)*>;

Summary of Optional Arguments

DATELINE
specifies that projections wrap from right to left at the international dateline.

DEGREES
specifies that the units for the longitude (X variable) and latitude (Y variable) coordinates are degrees.

DUPOK
specifies that observations are retained when their projected X and Y values are identical to those in the previous observation.

EASTLONG
specifies that the longitude (X variable) values in the input map data set increase to the east.

FROM="*coordinate-system*"
invokes a proj.4 projection and specifies a coordinate system from which to start the projection.

LATLON
specifies that the LAT and LONG variables from the map data set are used for coordinate data instead of the Y and X variables.

LATMAX=*max-latitude*
specifies the maximum latitude that is included in the projection.

LATMIN=*min-latitude*
: specifies the minimum latitude that is included in the projection.

LONGMAX=*max-longitude*
: specifies the maximum longitude to be included in the projection.

LONGMIN=*min-longitude*
: specifies the minimum longitude to be included in the projection.

MERIDIAN=*longitude-value*
: specifies the longitude value in degrees for the center of the projection.

NODATELINE
: enables contiguous projections for maps that cross the line between 180 degrees and -180 degrees longitude.

NODUP
: specifies that observations are deleted when their projected X and Y values are identical to those in the previous observation.

NOPARMIN
: specifies that parameters should not be used from the PROJPARM data set.

OUT=*output-map-data-set*
: names the output map data set.

PARADIV=*n*
: specifies the divisor that computes the values used for standard parallels for the Albers' or Lambert's projections when explicit values are not provided.

PARALLEL1=*latitude*

PARALLEL2=*latitude*
: specify values for the standard parallels that are used in the Albers' or Lambert's projection.

PARMENTRY=*entry-name*
: specifies the entry in the projection parameters data set that is used for input parameters.

PARMIN=*data-set*
: specifies a data set that contains input projection parameters.

PARMOUT=*data-set*

PARMOUT
: creates a data set that contains the projection parameters for the current PROC GPROJECT statement.

POLELAT=*latitude*

POLELONG=*longitude*
: each specifies a projection pole to use for the gnomonic projection.

PROJECT=ALBERS | LAMBERT | GNOMON | PROJ4 | NONE
: specifies the projection method to apply to the map data set.

RADIANS
: specifies that the units for the longitude (X variable) and latitude (Y variable) coordinates are radians.

TO="*coordinate-system*"
: invokes the proj.4 projection and specifies a coordinate system for the result of the transforming projection.

WESTLONG
: specifies that the longitude (X variable) values in the input map data set increase to the west.

Required Argument

DATA=*input-map-data-set*
identifies the map data set that you want to process. If not specified, an input map data set must have been created before issuing the PROC GPROJECT statement.

Default The GPROJECT procedure uses the most recently created SAS data set.

Requirement This statement is required if no input map data set was created before issuing the PROC GPROJECT statement.

See "About the Input Map Data Set" on page 357

"The SAS Data Set: Your Key to the SAS System" in *Step-by-Step Programming with Base SAS*

"About Data Set Options" in *SAS Data Set Options: Reference*

Example "Example 4: Projecting an Annotate Data Set" on page 384

Optional Arguments

DATELINE
specifies that projections wrap from right to left at the international dateline.

This option does not alter the longitude values that are provided in the input data, and assumes that these values are in the –180 to 180 degree range.

Default The DATELINE option is in effect by default. You can use DATELINE to override the NODATELINE option from an input parameters file.

Interaction If you specified both DATELINE and NODATELINE, then the last option that you specified is used.

DEGREES
specifies that the units for the longitude (X variable) and latitude (Y variable) coordinates are degrees. The GPROJECT procedure stops processing the data set if coordinates are out of range.

Alias DEG

Default Coordinate units are considered to be radians unless this option is specified.

Interactions If you specify both DEGREES and RADIANS, then the last option that you specified is used.

The DEGREES option is used by default with the proj.4 projection method.

DUPOK
specifies that observations are retained when their projected X and Y values are identical to those in the previous observation. By default, successive identical observations are deleted.

Alias ASIS

Interaction If you specify both NODUP and DUPOK, then the last option that you specified is used.

Note This option is useful when you want to add annotation to a map that contains duplicate coordinates.

EASTLONG

specifies that the longitude (X variable) values in the input map data set increase to the east (that is, positive longitude values are east of the prime meridian.)

Alias EAST

Default Longitude values increase to the west unless this option is specified.

Interactions If you specify both EASTLONG and WESTLONG, then the last option that you specified is used.

The EASTLONG option is used by default with the proj.4 projection method.

FROM="*coordinate-system*"

invokes a proj.4 projection and specifies a coordinate system from which to start the projection. By default, the proj.4 projection enables a transformation from latitude and longitude geographic coordinates (EPSG:4326) to a Mercator (900913) coordinate system. You can use the FROM= option to specify a transformation from a coordinate system different from EPSG:4326.

Requirement Quotation marks are required when specifying *"coordinate-system"*. You can specify *"coordinate-system"* as a shortcut name or as a fully expanded string that the proj.4 projection mode understands. For example, use either of the following two equivalent specifications:

```
FROM="900913"

FROM="+proj=merc +a=6378137 +b=6378137 +lat_ts=0.0
+lon_0=0.0 +x_0=0.0 +y_0=0 +k=1.0 +units=m
+nadgrids=@null +wktext  +no_defs"
```

The data set SASHELP.PROJ4DEF contains the possible *"coordinate-system"* values. Values from the EPSG, ESRI, and Mercator systems are included. Specifically, this data set contains shortcut names, the proj.4 strings that the shortcuts are translated into, and a short description of each coordinate-system. To view this data set, run the following code:

```
proc print data=sashelp.proj4def;
run;
```

Interaction Specify both the FROM= and TO= options to reverse a projection. For example, if you already have a Mercator projection, you can use the FROM= option in conjunction with the TO= option to revert the projection to EPSG:4326. You can also use the TO= option to specify a transformation to a coordinate system different from Mercator.

Note The Mercator coordinate system is used by Google and OpenStreetMap (OSM) in their maps.

See "TO="*coordinate-system*"" on page 372

LATLON

specifies that the LAT and LONG variables from the map data set are used for coordinate data instead of the Y and X variables. The LAT and LONG variables represent the unprojected latitude and longitude coordinates, respectively. All GfK GeoMarketing and most traditional map data sets contain both sets of variables (Y, X, and LAT, LONG) for projected and unprojected maps. By default, the Y and X variables are used to produce a projected map. However, when LATLON is specified, the Y and X variables are no longer required by the GPROJECT procedure statement.

Alias LATLONG

Example "Example 4: Projecting an Annotate Data Set" on page 384

LATMAX=*max-latitude*

specifies the maximum latitude that is included in the projection. Any unit areas that cross the selected latitude are clipped and closed along the specified parallels. The LATMAX= and LATMIN= options do not have to be paired; you can specify a maximum latitude without specifying a minimum.

When PROJECT=ALBERS, LAMBERT, GNOMON, or PROJ4, the GPROJECT procedure treats the value of *max-latitude* as degrees. When PROJECT=NONE, the procedure treats the value as a Cartesian coordinate.

Example "Example 3: Clipping an Area from the Map" on page 382

LATMIN=*min-latitude*

specifies the minimum latitude that is included in the projection. Any unit areas that cross the selected latitude are clipped and closed along the specified parallels. The LATMAX= and LATMIN= options do not have to be paired; you can specify a minimum latitude without specifying a maximum.

When PROJECT=ALBERS, LAMBERT, GNOMON, or PROJ4, the GPROJECT procedure treats the value of *min-latitude* as degrees. When PROJECT=NONE, the procedure treats the value as a Cartesian coordinate.

Example "Example 3: Clipping an Area from the Map" on page 382

LONGMAX=*max-longitude*

specifies the maximum longitude to be included in the projection. Any unit areas that cross the selected longitude are clipped and closed along the specified meridians. The LATMAX= and LATMIN= options do not have to be paired; you can specify a maximum longitude without specifying a minimum.

When PROJECT=ALBERS, LAMBERT, GNOMON, or PROJ4, the GPROJECT procedure treats the value of *max-longitude* as degrees. When PROJECT=NONE, the procedure treats the value as a Cartesian coordinate.

Example "Example 3: Clipping an Area from the Map" on page 382

LONGMIN=*min-longitude*

specifies the minimum longitude to be included in the projection. Any unit areas that cross the selected longitude are clipped and closed along the specified meridians. The LATMAX= and LATMIN= options do not have to be paired; you can specify a minimum longitude without specifying a maximum.

When PROJECT=ALBERS, LAMBERT, GNOMON, or PROJ4, the GPROJECT procedure treats the value of *min-longitude* as degrees. When PROJECT=NONE, the procedure treats the value as a Cartesian coordinate.

Example "Example 3: Clipping an Area from the Map" on page 382

MERIDIAN=*longitude-value*
specifies the longitude value in degrees for the center of the projection. By default, the meridian value is the center of the map data or clip region.

Note: MERIDIAN= is functionally equivalent to POLELONG=.

NODATELINE
enables contiguous projections for maps that cross the line between 180 degrees and -180 degrees longitude.

This option assumes that the longitude values that are provided in the input data are in the range of –180 to 180 degrees. The NODATELINE option adjusts the longitude values to a range of 0 to 360 degrees by adding 360 degrees to any value less than 0 degrees.

For example, if you project a map of Asia, then the eastern tip of the continent might be projected on the left side of the map by default. The NODATELINE option enables the entire continent to be projected as a contiguous area.

Restriction Before using this option with a map region that crosses the prime meridian—a longitudinal line of 0 degrees—you must adjust the data in your input map data set. Divide the region up into polygons drawn on either side of the prime meridian. Otherwise, the GMAP procedure, when using as input the projected output map data set, produces a map with an inaccurate, distorted appearance.

Interaction If you specified both DATELINE and NODATELINE, then the last option that you specified is used.

NODUP
specifies that observations are deleted when their projected X and Y values are identical to those in the previous observation.

Note: The NODUP option is in effect by default. You can use NODUP to override the DUPOK option from an input parameters file.

Alias NOASIS

Interaction If you specify both NODUP and DUPOK, then the last option that you specified is used.

NOPARMIN
specifies that parameters should not be used from the PROJPARM data set. By default, the GPROJECT procedure reads input parameters from the PROJPARM data set of the library that contains the input data set.

OUT=*output-map-data-set*
names the output map data set, which contains the coordinates of the new unit areas that are created by the GPROJECT procedure.

By default, the GPROJECT procedure names the new data set that uses the DATA*n* naming convention. That is, the procedure uses the name WORK.DATA*n*, where *n* is the next unused number in sequence. Thus, the first automatically named data set is DATA1, the second is DATA2, and so on.

Example "Example 4: Projecting an Annotate Data Set" on page 384

PARADIV=*n*
specifies the divisor that computes the values used for standard parallels for the Albers' or Lambert's projections when explicit values are not provided. By default PARADIV=4, which causes the standard parallels to be set at 1/4 and 3/4 of the range of latitude values in the input map data set.

See "PARALLEL1=*latitude*PARALLEL2=*latitude*" on page 370

PARMIN=*data-set*
specifies a data set that contains input projection parameters.

Alias PIN

Default The PROJPARM data set of the library that contains the input data set.

PARMOUT=*data-set*
PARMOUT
creates a data set that contains the projection parameters for the current PROC GPROJECT statement. If you specify a data set, then the parameters are created in that data set. If you do not specify a data set, then the parameters are created in PROJPARM data set of the library that contains the output data set.

Alias POUT

PARMENTRY=*entry-name*
specifies the entry in the projection parameters data set that is used for input parameters. By default, the entry with the same name as the input data set is used. For example, if you specify **`data=MYMAPS.MAP5`**, then the GPROJECT procedure uses parameters from an entry named MAP5 in the parameters data set.

Alias PENTRY

PARALLEL1=*latitude*
PARALLEL2=*latitude*
specify values for the standard parallels that are used in the Albers' or Lambert's projection. *Latitude* must be in degrees. Positive values indicate north of the equator, and negative values indicate south of the equator. These options are ignored for the gnomonic projection.

By default, the GPROJECT procedure calculates values for the standard parallels. The defaults are chosen to minimize the distortion inherent in the projection process. The algorithm used is as follows:

PARALLEL1 = minlat + R / P_D

PARALLEL2 = maxlat - R / P_D

where:

R
is the range of latitude values in the input map data set.

P_D
is the PARADIV= value. (See the discussion of the "PARADIV=*n*" on page 370 option.)

minlat
is the minimum latitude value in the input map data set.

maxlat
is the maximum latitude value in the input map data set.

If you do not use PARALLEL1= or PARALLEL2=, or you omit either option, the GPROJECT procedure uses the calculated value for the missing parameter.

The standard parallels, whether explicitly specified or supplied by the procedure, must lie on the same side of the equator. If they do not, PROC GPROJECT prints an error message and stops (the procedure might calculate standard parallels that lie on opposite sides of the equator). There might be cases where you are projecting a map data set that contains unit areas that cross the equator. You might have to explicitly specify standard parallels that both lie on the same side of the equator. If this causes excessive distortion of the map, you might be able to use the gnomonic projection instead of the Albers' or Lambert's projection. This is because the gnomonic technique has no such limitations at the equator.

Alias PARALEL1, PARALEL2

POLELAT=*latitude*
POLELONG=*longitude*
each specifies a projection pole to use for the gnomonic projection. The projection pole is the point at which the surface of the sphere touches the surface of the imaginary plane onto which the map is projected. The POLELAT= option specifies the latitude of the projection point.

Units for latitude are degrees; positive values indicate north of the equator, and negative values indicate south of the equator. The POLELONG= option gives the longitude for the projection point. Units for longitude are degrees; positive values indicate west of the prime meridian, and negative values indicate east of the prime meridian. This is true unless EASTLONG also has been used in the PROC GPROJECT statement.

You might choose not to use the POLELAT= or POLELONG= option, or you might omit either option. In these cases, PROC GPROJECT uses values for the position of the center of the unit areas that are defined by the DATA= data set for the missing parameter.

Note The map that is defined by the input map data set should not contain points more than 85 degrees (1.48353 radians) from the projection pole; all points that exceed this value are deleted from the output map data set.

Example "Example 2: Emphasizing Map Areas" on page 380

PROJECT=ALBERS | LAMBERT | GNOMON | PROJ4 | NONE
specifies the projection method to apply to the map data set. Values for the PROJECT= option are as follows:

ALBERS
specifies Albers' equal-area projection with two standard parallels.

LAMBERT
specifies Lambert's conformal projection with two standard parallels.

GNOMON
specifies the gnomonic projection, which is an azimuthal projection.

PROJ4
specifies a transformation projection, the default being from EPSG:4386 (LAT and LONG) to 900913 (Mercator) coordinate system. These defaults can be overridden with the FROM= and TO= options.

Default The DEGREES and EASTLONG options are enabled with the proj.4 projection method.

See "TO="*coordinate-system*"" on page 372

"FROM="*coordinate-system*"" on page 367

NONE
specifies that no projection should be performed. Use this option in conjunction with the LATMIN=, LATMAX=, LONGMIN=, and LONGMAX= options to perform clipping without projection.

By default, PROJECT=ALBERS.

Note There are several additional projections available. These projections are experimental and are not supported by SAS Institute Technical Support. The experimental projections are: ADAMS, AITOFF, APIANUS, ARAGO, BEHRMANN, BRAUN, CYLINDRI, ECKERT1, ECKERT3, ECKERT5, EQUIRECT, or MARINUS, GALL, HAMMER, KVRSKY7, MILLER1, MILLER2 , ORTHO, PARABOLI, PETERS, PUTNINS4, ROBINSON, STEREO, WINKEL2.

See "About Types of Map Projections" on page 359

Example "Example 2: Emphasizing Map Areas" on page 380

RADIANS
specifies that the units for the longitude (X variable) and latitude (Y variable) coordinates are radians.

Note: The RADIANS option is in effect by default. You can use RADIANS to override the DEGREES option from an input parameters file.

Alias NODEG

Interaction If you specify both RADIANS and DEGREES, then the last option that you specified is used.

TO="*coordinate-system*"
invokes the proj.4 projection and specifies a coordinate system for the result of the transforming projection. By default, the proj.4 projection enables a transformation from latitude and longitude geographic coordinates (EPSG:4326) to a Mercator (900913) coordinate system. You can use the TO= option to specify a transformation to a coordinate system different from Mercator.

Requirement Quotation marks are required when specifying *"coordinate-system"*. You can specify *"coordinate-system"* as a shortcut name or as a fully expanded string that the proj.4 projection mode understands. For example, use either of the following two equivalent specifications:

```
TO="EPSG:4326"

TO="+proj=longlat +datum=WGS84 +no_defs"
```

The data set SASHELP.PROJ4DEF contains the possible *"coordinate-system"* values. Values from the EPSG, ESRI, and Mercator systems are included. Specifically, this data set contains shortcut names, the proj.4 strings that the shortcuts are translated

into, and a short description of each coordinate-system. To view this data set, run the following code:

```
proc print data=sashelp.proj4def;
run;
```

Interaction Specify both the FROM= and TO= options to reverse a projection. For example, if you already have a Mercator projection, you can use the TO= option in conjunction with the FROM= option to revert the projection to EPSG:4326. You can also use the FROM= option to specify a transformation from a coordinate system different from EPSG:4326.

Note The Mercator coordinate system is used by Google and OpenStreetMap (OSM) in their maps.

See "FROM="*coordinate-system*"" on page 367

WESTLONG
specifies that the longitude (X variable) values in the input map data set increase to the west (that is, positive longitude values are west of the prime meridian.)

Note: The WESTLONG option is in effect by default. You can use WESTLONG to override the EASTLONG option from an input parameters file.

Alias WEST

Interaction If you specify both WESTLONG and EASTLONG, then the last option that you specified is used.

ID Statement

Identifies the variable or variables that define the hierarchy of the current unit areas in the input map data set.

See: "Example 1: Using Default Projection Specifications" on page 376

Syntax

ID <*id-variable(s)*>;

Optional Argument

id-variable(s)
specifies one or more variables in the input map data set that identify unit areas. *Id-variable* can be either numeric or character.

Each group of observations with a different ID variable value is evaluated as a separate unit area.

Details

TIP You can use the ID statement without specifying an ID variable to project non-polygonal data (for example, annotation points.)

Using the GPROJECT Procedure

About the GPROJECT Procedure

You can use PROC GPROJECT statement options to do the following actions:

- select the map projection method
- specify the map projection criteria
- create a rectangular subset of the input map data set

The following sections describe how you can use PROC GPROJECT statement options to select your own projection method and projection criteria.

Selecting Projections

Except when projecting map data sets that cover large areas, all three types of projections (Albers', Lambert's, and gnomonic) produce relatively similar results when you use default projection criteria. Therefore, you usually do not need to be concerned about which projection method to use when you produce maps of small regions.

However, the default projection criteria might be unsuitable in some circumstances. In particular, the default specifications fail when the map that is being projected extends on both sides of the equator. On other occasions, you might want to select a projection method to achieve a particular effect.

For the Albers' and Lambert's projections, the two standard parallels must both lie on the same side of the equator. PROC GPROJECT stops and gives an error message if this condition is not met, regardless of whether you explicitly specify parallel values or let the procedure calculate default values. See the descriptions of the PARALLEL1= and PARALLEL2= options on page 370 for more information about how to specify the two standard parallels.

Controlling Projection Criteria

For both the Albers' and Lambert's projections, PROC GPROJECT calculates appropriate standard parallels. You can override either or both of these selections if you explicitly specify values for the PARALLEL1= or PARALLEL2= option. You can influence the selection of default parallels if you use the PARADIV= option.

For the gnomonic projection, PROC GPROJECT determines the longitude and latitude of the approximate center of the input map data set area. You can override either or both of these selections if you explicitly specify values for the POLELAT= or POLELONG= option.

The clipping options, discussed in “Clipping Map Data Sets” on page 375, can also influence the calculations of the default standard parallels by changing the minimum and maximum coordinate values.

Clipping Map Data Sets

The GPROJECT procedure can create rectangular subsets of the input map data set. This capability provides a way to extract a portion of a larger map if you do not need all the original unit areas for your graph. The procedure enables you to clip unit area boundaries at specified latitude values, longitude values, or both. Unit areas that fall completely outside of the specified clipping limits are excluded from the output map data set. Unit areas bisected by the clipping limits are closed along the clipping parallels and meridians, and all points outside of the clipping limits are excluded.

If the input map data set contains the DENSITY variable, any new vertex points and corners that are created by PROC GPROJECT are assigned a DENSITY value of 0 in the output map data set. This enables you to use a subset of the clipped map without using PROC GREDUCE to assign new DENSITY values. (See Chapter 9, "GREDUCE Procedure," on page 389 for information about how to reduce the number of points that you need to draw a map.)

You can specify the minimum latitude to be retained in the output map data set with the LATMIN= option and the maximum latitude with LATMAX= option. Minimum and maximum longitude values are specified with the LONGMIN= and LONGMAX= options, respectively.

This is how the PROC GPROJECT interprets the clipping longitude and latitude values:

- If you specify PROJECT=NONE in the PROC GPROJECT statement, the procedure assumes that the input map data set is already projected and the clipping longitude and latitude values are Cartesian coordinates. In this case, the LATMAX= and LATMIN= options specify the top and bottom edges, respectively, of the area that you want to extract, and the LONGMAX= and LONGMIN= options specify right and left edges, respectively.

  You must be familiar with the range of values in the X and Y variables in order to select appropriate clipping limits. Use the MEANS or SUMMARY procedure in Base SAS to determine the range of values in X and Y. See the *Base SAS Procedures Guide* for more information.

- In all other projections, the clipping values are treated as degrees.

Depending on the size and position of the clipped area and the type of projection that is performed, the resulting map might not be exactly rectangular. PROC GPROJECT performs clipping before projection, so the clipped area might be distorted by the projection process.

To produce a clipped area with a rectangular shape, use PROC GPROJECT in two steps:

1. Project the map using the appropriate projection method and projection criteria.
2. Project the map using PROJECT=NONE, and use the LATMIN=, LATMAX=, LONGMIN=, and LONGMAX= options to clip the map.

See "Example 3: Clipping an Area from the Map" on page 382, for an example of clipping an area from an unprojected map data set.

Examples: GPROJECT Procedure

Example 1: Using Default Projection Specifications

Features: PROC GPROJECT options
DATA= required argument
OUT= optional argument
ID statement
PATTERN statement

Other features: DATA step
PROC GMAP statement
CHORO statement

Data sets: MAPS.STATES (input map data set for DATA step)
US48PROJ (input map data set and map data set)

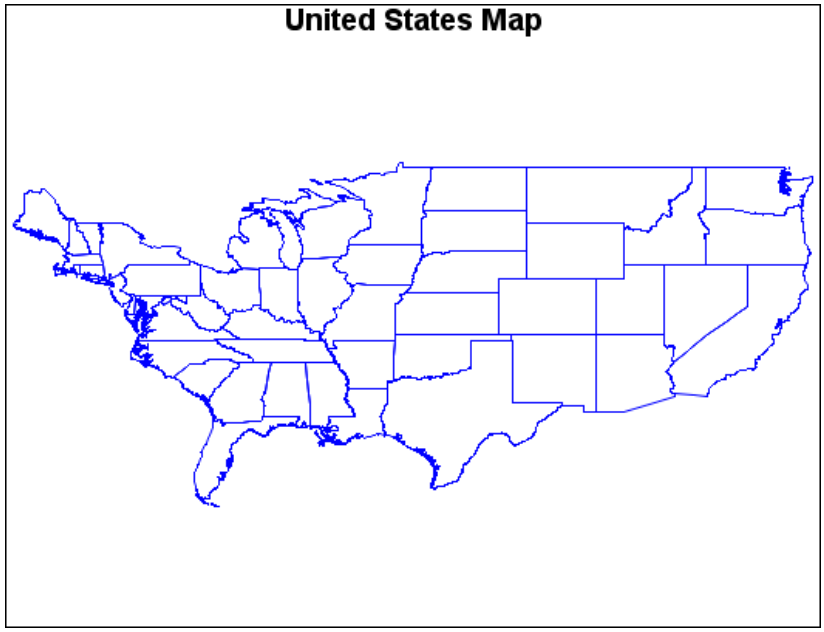
Sample library member: GPJDEFLT

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

This example demonstrates the effect of using PROC GPROJECT on an unprojected map data set without specifying any options. Because the PROJECT= option is not used in the PROC GPROJECT statement, the Albers' equal-area projection method is used by default. PROC GPROJECT supplies defaults for the standard parallels that minimize the distortion of the projected map areas.

Output 8.1 *Map before Projection (GPJDEFLT(a))*



Output 8.1 on page 377 illustrates the output produced by the US48 map data set, which contains unprojected values in the X and Y variables. The X values are measured in westlong degrees. Note that in GfK map data sets the unprojected longitude (LONG) coordinate values are measured in eastlong degrees. Using the GPROJECT procedure with GfK values would result in a map with the east coast on the right side. Output 8.2 on page 377 shows the variables in the data set.

Output 8.2 *The US48 Data Set*

US48 Data Set

OBS	STATE	SEGMENT	DENSITY	X	Y
1	1	1	3	1.48221	0.56286
2	1	1	3	1.48226	0.56234
3	1	1	3	1.48304	0.56231
.					
.					
.					

The GPROJECT procedure is used with the US48 map data set as input to create the projected map data set, US48PROJ. The values for X and Y in this new data set are projected (Cartesian). Output 8.3 on page 378 shows the variables in the data set.

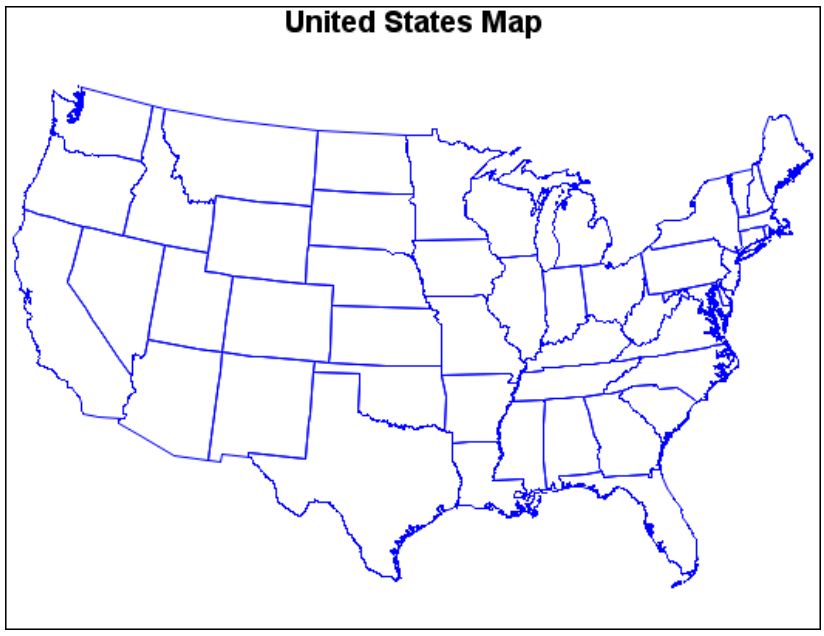
Output 8.3 *The US48PROJ Data Set*

US48PROJ Data Set

OBS	X	Y	DENSITY	STATE	SEGMENT
1	0.16068	-0.073470	3	1	1
2	0.16069	-0.073993	3	1	1
3	0.16004	-0.074097	3	1	1
.					
.					
.					

The new projected map data set, US48PROJ, is used to create the projected map, Output 8.4 on page 378.

Output 8.4 *Map after Projection (GPJDEFLT(b))*



Program

```
goptions reset=all border;

data us48;
   set maps.states;
   if state ne 2 and state ne 15 and state ne 72;
run;

title "United States Map";

pattern value=mempty color=blue;

proc gmap map=us48 data=us48 all density=4;
   id state;
   choro state / nolegend levels=1;
run;

proc gproject data=us48
              out=us48proj;
```

```
   id state;
run;

proc gmap map=us48proj
          data=us48proj all density=4;
   id state;
   choro state / nolegend levels=1;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create a reduced continental U.S. map data set and remove Alaska, Hawaii, and Puerto Rico.

```
data us48;
   set maps.states;
   if state ne 2 and state ne 15 and state ne 72;
run;
```

Define the title for the unprojected map.

```
title "United States Map";
```

Define the pattern characteristics.

```
pattern value=mempty color=blue;
```

Show the unprojected map.

```
proc gmap map=us48 data=us48 all density=4;
   id state;
   choro state / nolegend levels=1;
run;
```

Project the map data set using all default criteria. The ID statement identifies the variable in the input map data set that defines unit areas.

```
proc gproject data=us48
              out=us48proj;
   id state;
run;
```

Show the projected map.

```
proc gmap map=us48proj
          data=us48proj all density=4;
   id state;
   choro state / nolegend levels=1;
run;
quit;
```

Example 2: Emphasizing Map Areas

Features: PROC GPROJECT options
- DATA= required argument
- OUT= optional argument
- POLELAT=
- POLELONG=
- PROJECT=

ID statement

PATTERN statement

Other features: DATA step

PROC GMAP statement

CHORO statement

Data sets: MAPS.STATES (input map data set for DATA step)

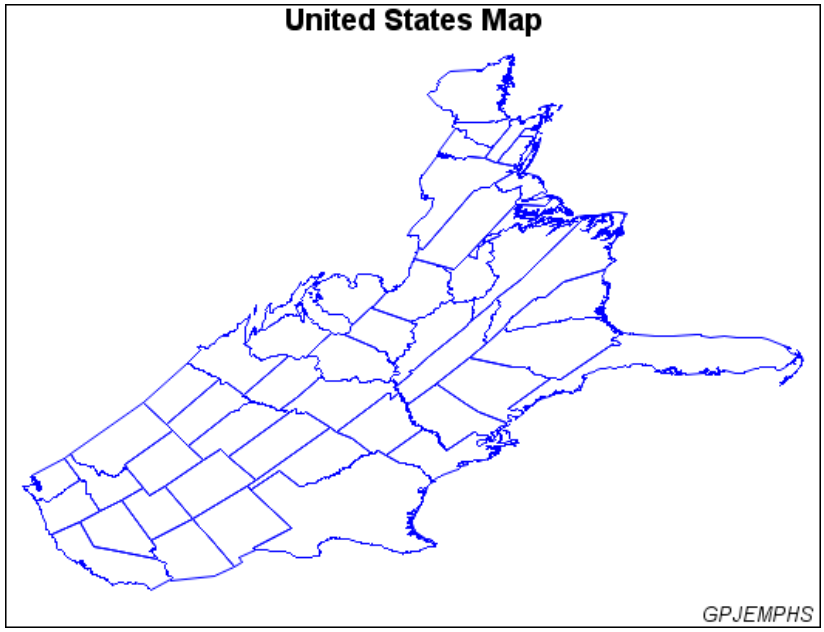
SKEW (input map data set and map data set)

Sample library member: GPJEMPHS

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

This example uses the gnomonic projection method to create a map in which the east coast of the United States appears disproportionately large compared to the west coast.



Program

```
goptions reset=all border;
```

```
data us48;
   set maps.states;
   if state ne 2 and state ne 15 and state ne 72;
   if density<4;
run;

proc gproject data=us48
              out=skew
              project=gnomon
              polelong=160
              polelat=45;
   id state;
run;

title "United States Map";
footnote j=r "GPJEMPHS ";

pattern value=mempty color=blue;

proc gmap map=skew data=skew all;
   id state;
   choro state / nolegend levels=1;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create a reduced continental U.S. map data set and remove Alaska, Hawaii, and Puerto Rico.

```
data us48;
   set maps.states;
   if state ne 2 and state ne 15 and state ne 72;
   if density<4;
run;
```

Project the map onto a plane centered in the Pacific. The PROJECT= option specifies the projection method for the map data set. The POLELONG= and POLELAT= option specify a projection pole for the gnomonic projection. In this example, the pole is positioned in the Pacific Ocean.

```
proc gproject data=us48
              out=skew
              project=gnomon
              polelong=160
              polelat=45;
   id state;
run;
```

Define the title and footnote for the map.

```
title "United States Map";
footnote j=r "GPJEMPHS ";
```

Define the pattern characteristics.

```
pattern value=mempty color=blue;
```

Show the projected map.

```
proc gmap map=skew data=skew all;
   id state;
   choro state / nolegend levels=1;
run;
quit;
```

Example 3: Clipping an Area from the Map

Features: PROC GPROJECT options
- DATA= required argument
- OUT= optional argument
- LONGMAX=
- LONGMIN=
- LATMAX=
- LATMIN=

ID statement

PATTERN statement

Other features: PROC GMAP statement

CHORO statement

Data sets: MAPS.STATES (input map data set for DATA step)

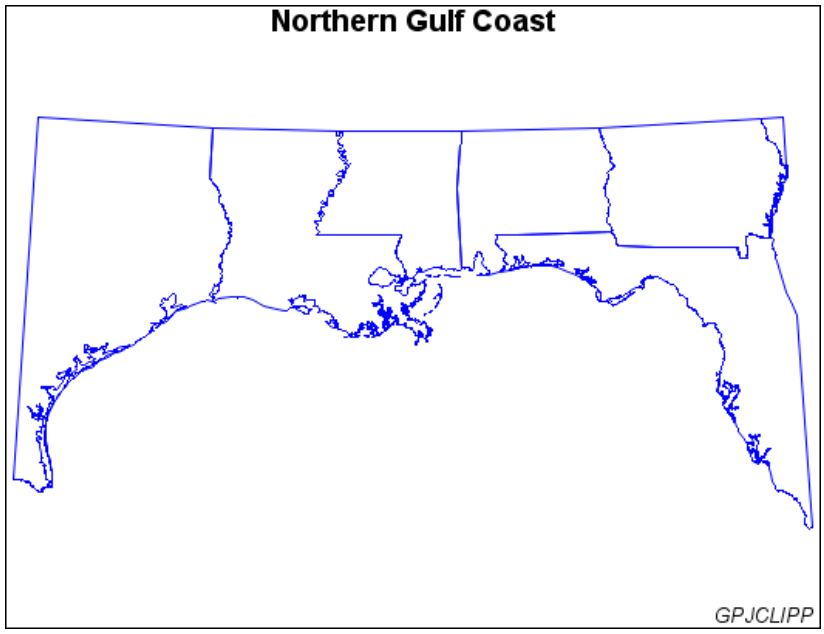
GULF (input map data set and map data set)

Sample library member: GPJCLIPP

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

This example uses the clipping capabilities of PROC GPROJECT to create a map of the states in the United States that border the Gulf of Mexico. Because the PROJECT= option is not used in the GPROJECT procedure, the Albers' equal-area projection method is used by default.



Program

```
goptions reset=all border;

proc gproject data=maps.states
              out=gulf
              longmin=81
              longmax=98
              latmin=25
              latmax=33;
   where density<5;
   id state;
run;

title "Northern Gulf Coast";
footnote j=r "GPJCLIPP ";

pattern value=mempty color=blue;

proc gmap map=gulf data=gulf all;
   id state;
   choro state / nolegend levels=1;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Clip and project a rectangular subset of the map. The LONGMIN= and LONGMAX= options specify the minimum and maximum longitudes to be included in the map projection. The LATMIN= and LATMAX= options specify the minimum and maximum latitudes to be included in the map projection.

```
proc gproject data=maps.states
```

```
                out=gulf
                longmin=81
                longmax=98
                latmin=25
                latmax=33;
   where density<5;
   id state;
run;
```

Define the title and footnote for the map.

```
title "Northern Gulf Coast";
footnote j=r "GPJCLIPP ";
```

Define the pattern characteristics.

```
pattern value=mempty color=blue;
```

Show the clipped map.

```
proc gmap map=gulf data=gulf all;
   id state;
   choro state / nolegend levels=1;
run;
quit;
```

Example 4: Projecting an Annotate Data Set

Features: PROC GPROJECT options
- DATA= required argument
- DEGREES
- LATLON
- OUT= optional argument
- PROJECT=
- POLELONG
- POLELAT
- PARMENTRY= optional argument
- PARMOUT
- DUPOK

ID statement

PATTERN statement

Other features: DATA step

PROC GMAP statement

CHORO statement

Annotate data set

Data sets: MAPSSAS.STATES (input map data set for DATA step)

MAPSSAS.USCITY (input map data set for DATA step)

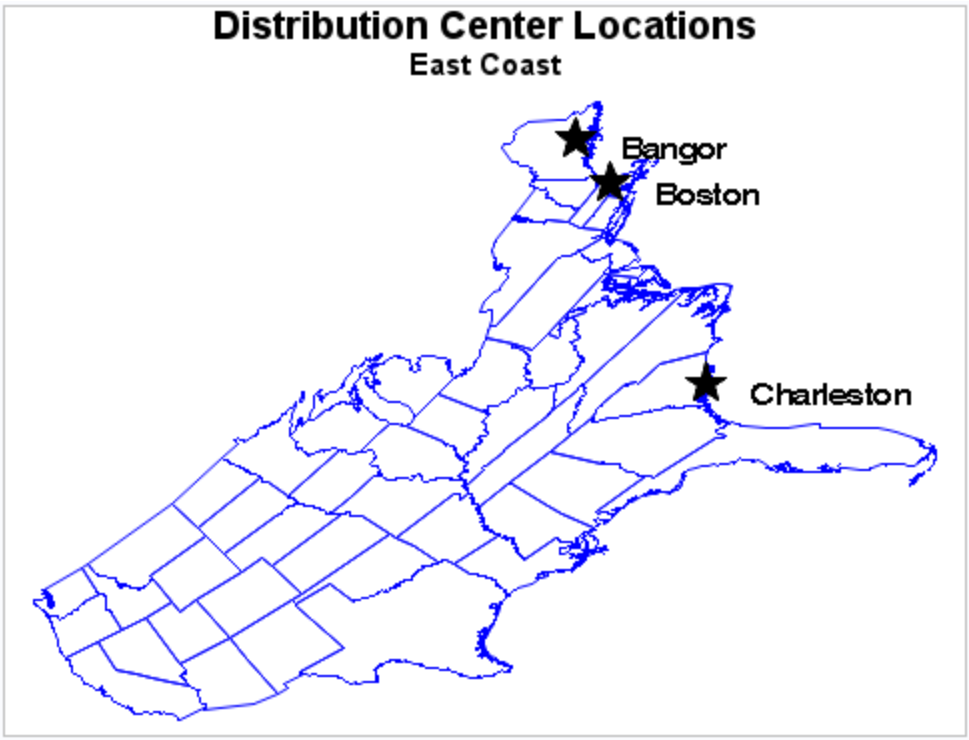
US48P (input map data set and map data set)

Sample library member: GPJANNOT

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

This example illustrates how to project an Annotate data set for use with a map data set. It labels the locations of Charleston, Boston, and Bangor on the map shown in the second example. Note that the Y and X variables in the MAPSSAS.USCITY data set have already been projected to match the US data set. Therefore, they cannot be used with the map that is produced by the second example. Instead, the LATLON option signals the GPROJECT procedure to use the unprojected LAT and LONG coordinate variables. To properly label the projected map, the example uses the same projection method for the city coordinates that is used for the map coordinates. This example illustrates how to use the same projection method for both data sets.



Program

```
goptions reset=all border;

data us48;
   set mapssas.states;
   if state ne 2 and state ne 15 and state ne 72;
   if density<4;
run;


proc gproject data=us48
              out=us48p
              project=gnomon
              polelong=160
              polelat=45
```

```
              parmout;
id state;
run;

data cities;
   set mapssas.uscity(keep=lat long city state);
   length function style color $ 8
          position $ 1 text $ 20;
   retain function "label" xsys ysys "2"
          hsys "1" when "a";
   if (state=45 and city="Charleston") or
      (state=25 and city="Boston") or
      (state=23 and city="Bangor");
    state+100; color="black"; size=8;
      text="V";
      position="5";
      style="marker";
      output;
    state+1; color="black"; size=5;
      text="   "||city;
      position="6";
      style="swissb";
      output;
run;

proc gproject data=cities
              degrees
              latlon
              out=citiesp
              parmentry=us48p
              dupok;
id;
run;

title1 "Distribution Center Locations";
title2 "East Coast";

pattern value=mempty color=blue;

proc gmap data=us48p map=us48p all;
   id state;
   choro state / nolegend levels=1 stat=first
                 annotate=citiesp;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create a reduced continental U.S. map data set and remove Alaska, Hawaii, and Puerto Rico.

```
data us48;
   set mapssas.states;
   if state ne 2 and state ne 15 and state ne 72;
```

```
   if density<4;
run;
```

Project the US48 data set. The DATA= option specifies the data set to be projected. The OUT= option specifies the name of the new projected data set that is created. The PROJECT= option specifies that PROC GPROJECT use the GNOMON projection method for the city coordinates. This is the projection method used for both the city coordinates and the map coordinates. The POLELONG and POLELAT options each specify a projection pole to use for the gnomonic projection. The PARMOUT creates the projection parameters data set. The ID statement identifies the variable in the input map data set that defines the U.S. state map areas.

```
proc gproject data=us48
              out=us48p
              project=gnomon
              polelong=160
              polelat=45
              parmout;
id state;
run;
```

Create the Annotate data set CITIES from the MAPSSAS.USCITY data set. The unprojected LONG and LAT variable values are kept for use instead of the projected X and Y variable values. The value of STATE is modified for each label to ensure that it is unique.

```
data cities;
   set mapssas.uscity(keep=lat long city state);
   length function style color $ 8
          position $ 1 text $ 20;
   retain function "label" xsys ysys "2"
          hsys "1" when "a";
   if (state=45 and city="Charleston") or
      (state=25 and city="Boston") or
      (state=23 and city="Bangor");
    state+100; color="black"; size=8;
      text="V";
      position="5";
      style="marker";
      output;
    state+1; color="black"; size=5;
      text="   "||city;
      position="6";
      style="swissb";
      output;
run;
```

Project the annotate data set. The DATA= option specifies the data set to be projected. The DEGREES option specifies that the coordinates are in degrees rather than radians. The LATLON option specifies that the unprojected LAT and LONG variables from the map data set are used for coordinate data instead of the projected Y and X variables. The OUT= option specifies the name of the new projected data set that is created. The PARMENTRY= option specifies the entry in the projection parameters data set that is used. The empty ID statement specifies that the data is not polygonal. Note that the PARMIN= option is not needed because the WORK.PROJPARM data set is used by default.

```
proc gproject data=cities
              degrees
              latlon
              out=citiesp
              parmentry=us48p
              dupok;
id;
run;
```

Define the titles for the map.

```
title1 "Distribution Center Locations";
title2 "East Coast";
```

Define the pattern characteristics.

```
pattern value=mempty color=blue;
```

Display the annotated map. The CHORO statement displays the projected map and annotates it using the projected Annotate data set.

```
proc gmap data=us48p map=us48p all;
   id state;
   choro state / nolegend levels=1 stat=first
                 annotate=citiesp;
run;
quit;
```

References

Pearson, F., II. 1977. *Map Projection Equations (Report Number TR-3624)*. Dahlgren, VA: Naval Surface Weapons Center, Dahlgren Laboratory.

Richardus, P. and Adler, R.K. 1972. *Map Projections*. Amsterdam, Netherlands: North-Holland Publishing Company; New York: American Elsevier Publishing Company.

Robinson, A.H. 1978. *Elements of Cartography*. New York, USA: John Wiley & Sons, Inc.

Chapter 9

GREDUCE Procedure

Overview: GREDUCE Procedure

The GREDUCE procedure processes map data sets so that they can draw simpler maps with fewer boundary points. It creates an output map data set that contains all of the variables in the input map data set plus a new variable named DENSITY. For each observation in the input map data set, the procedure determines the significance of that point for maintaining a semblance of the original shape. It also gives the observation a corresponding DENSITY value.

You can then use the value of the DENSITY variable to create a subset of the original map data set. The observations in the subset can draw a map that retains the overall appearance of the original map. However, the difference is that it contains fewer points, requires considerably less storage space, and can be drawn much more quickly.

GREDUCE does not produce any graphics output. Instead, it produces an output map data set that can become either one of the following:

- the input map data set for the GMAP procedure
- the input map data set for a DATA step that removes points from the map

Figure 9.1 on page 390 and Figure 9.2 on page 390 illustrate the effect of reduction on a typical map data set. Figure 9.1 on page 390 uses observations with all DENSITY values as input to the GMAP procedure.

Figure 9.1 *CANADA2 Map before Reduction*

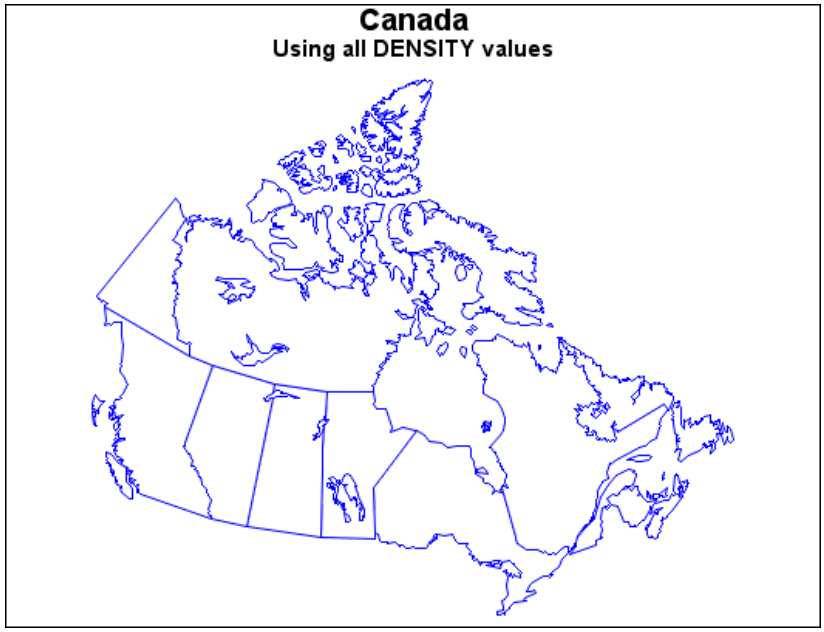
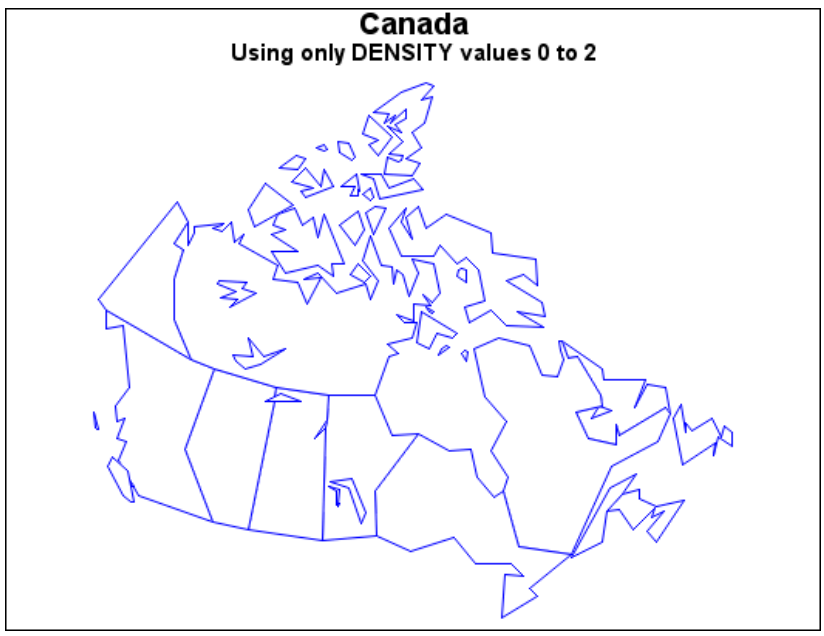


Figure 9.2 on page 390 uses only those observations with a DENSITY value of 0 to 2 as input to the GMAP procedure.

Figure 9.2 *CANADA2 Map after Reduction*



The program for these maps is in "Example: Reducing the Map of Canada" on page 397.

The reduced map shown in Figure 9.2 on page 390 retains the overall shape of the original but requires only 463 observations compared to the 4302 observations that are needed to produce the map in Figure 9.1 on page 390.

Note: Many of the map data sets that are supplied by SAS Institute already have been processed by GREDUCE. If the map data set contains a DENSITY variable, you do not need to process the data set using GREDUCE.

See also Chapter 10, "GREMOVE Procedure," on page 401 for more information about how to do the following:

- combine groups of unit areas into larger unit areas to create regional maps
- remove some of the boundaries in a map and create a subset of a map that combines the original areas

About the Input Map Data Set

The input map data set must be a SAS data set and contain these variables:

- a numeric variable named X that contains the horizontal coordinates of the map boundary points.
- a numeric variable named Y that contains the vertical coordinates of the map boundary points.
- one or more identification variables that uniquely identify the unit areas in the map. These variables are listed in the ID statement.

It can also contain the following:

- one or more variables that identify groups of unit areas (for BY-group processing)
- the variable SEGMENT, which distinguishes nonconterminous segments of the unit areas.

Any other variables in the input map data set do not affect the GREDUCE procedure.

About Unmatched Area Boundaries

If you are using map data sets in which area boundaries do not match precisely (for example, if the boundaries were digitized with a different set of points), PROC GREDUCE is unable to identify common boundaries properly. This results in abnormalities in your maps. These abnormalities include mismatched borders, missing vertex points, stray lines, gaps, and distorted polygons.

If the points in the area boundaries match up except for precision differences, round each X and Y value in your map data set accordingly. To accomplish this use the DATA step function ROUND before using PROC GREDUCE. (See *SAS Functions and CALL Routines: Reference* for information about the ROUND function.)

Take, for example, the map data set APPROX, which has horizontal and vertical coordinate values for interior boundaries of unit areas that are exactly equal only to three decimal places. The following DATA step creates a new map data set, EXACT, that is better suited for use with the GREDUCE procedure:

```
data exact;
      set approx;
      if x ne . then x=round(x,.001);
      if y ne . then y=round(y,.001);
```

```
run;
```

See "About Map Data Sets" on page 193 for additional information about map data sets.

Syntax: GREDUCE Procedure

Requirement: Exactly one ID statement is required.

Note: The procedure can include the "BY Statement" on page 19.

PROC GREDUCE DATA=*input-map-data-set* <*option(s)*>;
ID *id-variable(s)*;

PROC GREDUCE Statement

Identifies the input and output map data sets. Can specify the reduction criteria.

Requirement: An input map data set is required. If not specified, an input map data set must have been created before issuing the PROC GREDUCE statement.

Syntax

PROC GREDUCE DATA=*input-map-data-set* <*option(s)*>;

Summary of Optional Arguments

E1=*min-distance*
E2=*min-distance*
E3=*min-distance*
E4=*min-distance*
E5=*min-distance*
: specify the minimum distance that a point must lie from a straight line segment to be included at density level 1, 2, 3, 4, or 5, respectively.

LATLON
: specifies that the LAT and LONG variables from the map data set are used for coordinate data instead of the Y and X variables.

N1=*max-points*
N2=*max-points*
N3=*max-points*
N4=*max-points*
N5=*max-points*
: specify that for density level 1, 2, 3, 4, or 5, the boundary of a unit area should contain no more than *max-points* points.

NOCLEAN
: disables post-processing cleanup of the output data set.

OUT=*output-data-set*
: names the new map data set, which contains all of the observations and variables in the original map data set plus the new DENSITY variable.

Required Argument

DATA=*input-map-data-set*
identifies the map data set that you want to process. If not specified, an input map data set must have been created before issuing the PROC GREDUCE statement.

Default The GREDUCE procedure uses the most recently created SAS data set.

Requirement This statement is required if no response data set was created before issuing the PROC GREDUCE statement.

See "About the Input Map Data Set" on page 391

"The SAS Data Set: Your Key to the SAS System" in *Step-by-Step Programming with Base SAS*

"About Data Set Options" in *SAS Data Set Options: Reference*

Optional Arguments

E1=*min-distance*
E2=*min-distance*
E3=*min-distance*
E4=*min-distance*
E5=*min-distance*
specify the minimum distance that a point must lie from a straight line segment to be included at density level 1, 2, 3, 4, or 5, respectively. That is, in a reduced curve of three points, the middle point is at least a distance that is *min-distance* from a straight line between the two outside points.

Express *min-distance* values in the units for the coordinate system of the input map data set. For example, if the input map data set contains coordinates that are expressed in radians, express the *min-distance* values in radians.

Specify the E*n*= values in decreasing order. For example, the E2= value should be less than the E1= value, and so on.

LATLON
specifies that the LAT and LONG variables from the map data set are used for coordinate data instead of the Y and X variables. The LAT and LONG variables represent the unprojected latitude and longitude coordinates, respectively. All GfK Marketing and most traditional map data sets contain both sets of variables (Y, X, and LAT, LONG) for projected and unprojected maps. By default, the Y and X variables are used to produce an output data set. However, when LATLON is specified, the Y and X variables are no longer required by the GREDUCE procedure statement.

Alias LATLONG

Example "Example 4: Projecting an Annotate Data Set" on page 384

N1=*max-points*
N2=*max-points*
N3=*max-points*
N4=*max-points*
N5=*max-points*
specify that for density level 1, 2, 3, 4, or 5, the boundary of a unit area should contain no more than *max-points* points.

Specify the N*n*= values in increasing order. For example, the N2= value should be greater than or equal to the N1= value, and so on.

By default, if you omit N*n*= and E*n* = , the GREDUCE procedure calculates values for the five N*n* = parameters using this formula:

$$Nn = n^2 \times N_{max}/36$$

Here N_{max} is the maximum number of points in any unit area in the input map data set. However, the restriction that the number of points for any level cannot be less than the number of points in level 0 still applies.

NOCLEAN
disables post-processing cleanup of the output data set. By default, the GREDUCE procedure post-processes the output data set to ensure that at least three vertices are present for each polygon at each density level. If you specify NOCLEAN, then shapes with only one or two points are included in the output data set.

OUT=*output-data-set*
names the new map data set, which contains all of the observations and variables in the original map data set plus the new DENSITY variable. If the input map data set contains a variable named DENSITY, the GREDUCE procedure replaces the values of the variable in the output map data set. The original values of the DENSITY variable from the input map data set are not included in the output map data set.

By default, the GREDUCE procedure names the new data set that uses the DATA*n* naming convention. That is, the procedure uses the name WORK.DATA*n*, where *n* is the next unused number in sequence. Thus, the first automatically named data set is DATA1, the second is DATA2, and so on.

ID Statement

Identifies the variable or variables that define the hierarchy of the current unit areas in the input map data set.

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

See: "Example: Reducing the Map of Canada" on page 397

Syntax

ID *id-variable(s)*;

Required Argument

id-variable(s)
specifies one or more variables in the input map data set that identify unit areas. *Id-variable(s)* can be either numeric or character.

Each group of observations with a different ID variable value is evaluated as a separate unit area.

Using the GREDUCE Procedure

Specifying Density Levels

GREDUCE uses default criteria for determining the appropriate DENSITY variable value for each observation in the input map data set. If you do not want to use the default criteria, use PROC GREDUCE options to select the following:

- the maximum number of observations for each DENSITY level
- the minimum distance that an intermediate point must lie from a line between two end points to be included in the level

If you do not explicitly specify criteria, the procedure computes and uses default values.

GREDUCE creates seven density levels, numbered 0 through 6. Specify criteria for density levels 1 through 5. You cannot define criteria for level 0. Level 0 is reserved for map vertex points, such as common corners of unit areas. You also cannot define criteria for level 6. Level 6 is assigned to those points that do not meet the criteria for any lower level.

Specify the maximum number of observations per density level using N*n*= in the PROC GREDUCE statement, and specify the minimum point distance using E*n*= . You must have knowledge of the X and Y variable values in the particular input map data set to determine appropriate values for E*n*=. See the En= option on page 393 for details.

Figure 9.3 on page 395 illustrates how to use the minimum distance parameter to determine which points belong in a particular density level. At density level *n*, only point C lies at a distance greater than the E*n*= value (70) from a line between points A and B. Thus, after reduction only point C remains between points A and B at density level *n*, and the resulting reduced boundary is shown in Figure 9.4 on page 396. See Douglas and Peucker (1973) for details of the algorithm used.

Figure 9.3 *Points in Data Set before Reduction*

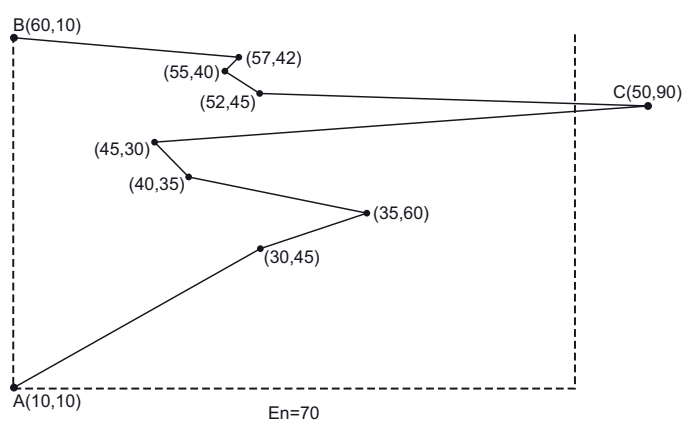
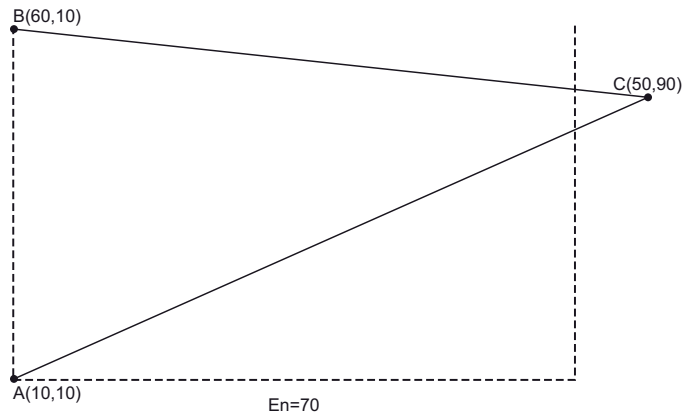


Figure 9.4 *Points in Data Set at Density n after Reduction*



GREDUCE uses the usual Euclidean distance formula to determine the distance between points. For example, the distance d between the points (x_0,y_0) and (x_1,y_1) is

$$d = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$$

This distance function might not suitable for the coordinate system in your input map data set. In that case, transform the X and Y values to an appropriate coordinate system before using GREDUCE. An example of inappropriate coordinates is latitude and longitude values around one of the poles. In this case, the data values should be projected before they are reduced. See Chapter 8, "GPROJECT Procedure," on page 355 for more information about map projection.

If you specify both N*n*= and E*n*= values for a density level, GREDUCE attempts to satisfy both criteria. However, the number of points for any level is never reduced below the number of points in density level 0. If you specify a combination of N*n*= or E*n*= values such that the resulting DENSITY values are not in order of increasing density, a note is printed in the SAS log, and the DENSITY values are calculated in increasing order of density.

Subsetting a Map Data Set

A map data set that is processed by GREDUCE does not automatically result in a map that uses fewer points. By default, the GMAP procedure produces a map that uses all of the points in the map data set. This occurs even if the data set has been processed by the GREDUCE procedure. To decrease the number of points that produce the map, you must create a subset of the original data set using a DATA step or the WHERE= data set option. For example, to create a subset of a map that uses only the DENSITY values 0, 1, and 2, use this DATA step:

```
data smallmap;
   set map;
   if density <= 2;
run;
```

Alternatively, you can use WHERE= in the PROC GMAP statement:

```
proc gmap map=map(where=(density<=2))
          data=response;
```

Note: GREDUCE does not reduce the size of the output map data set compared to the input map data set. By default, the output map data set from PROC GREDUCE is larger than the input map data set. This is because it contains all of the variables and

observations from the original data set, along with the DENSITY variable if it was not present in the original data set. If the input map data set already had a DENSITY variable, the output map data set is the same size as the input map data set.

Example: Reducing the Map of Canada

Features: ID statement

Sample library member: GRDCANAD

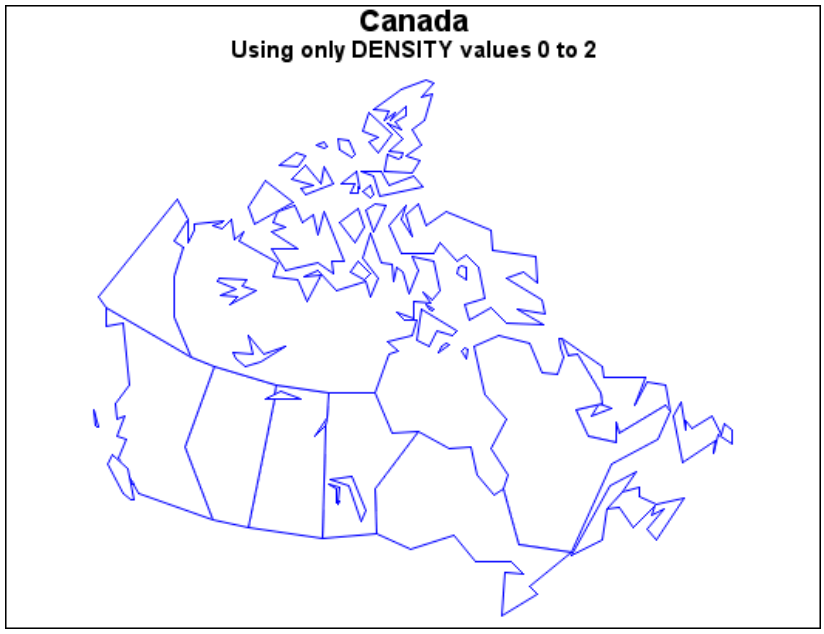
Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

This example illustrates major features of the GREDUCE procedure. Because the example uses one of the map data sets that are supplied with SAS/GRAPH, you might need to replace *SAS-data-library* in the LIBNAME statement with the actual location of the SAS library that contains the Institute-supplied map data sets on your system. If your site automatically assigns the libref MAPS to the SAS library that contains the Institute-supplied map data sets, delete the LIBNAME statement in this example.

In this example, the GREDUCE procedure creates the DENSITY variable for the CANADA2 map data set that is provided with SAS/GRAPH. First, the map is displayed at its original density by using the GMAP procedure.

Second, the map is displayed by using density values of 0 to 2.



Program

```
goptions reset=all border;

title1 "Canada";
title2 "Using all DENSITY values";

pattern value=msolid repeat=12 color=white;

proc gmap map=maps.canada2 data=maps.canada2 all;
  id province;
  choro province / nolegend coutline=blue;
run;

proc greduce data=maps.canada2 out=can2;
   id province;
run;

title2 "Using only DENSITY values 0 to 2";

proc gmap map=can2
          data=can2 all density=2;
   id province;
   choro province / nolegend coutline=blue;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Define the title for the first map.

```
title1 "Canada";
title2 "Using all DENSITY values";
```

Define pattern characteristics.

```
pattern value=msolid repeat=12 color=white;
```

Show the unreduced map. The ID statement specifies the variable in the map data set that defines unit areas.

```
proc gmap map=maps.canada2 data=maps.canada2 all;
  id province;
  choro province / nolegend coutline=blue;
run;
```

The GREDUCE procedure creates a new map data set, CAN2, containing a DENSITY variable. The ID statement specifies the variable in the map data set that defines unit areas.

```
proc greduce data=maps.canada2 out=can2;
   id province;
run;
```

Define the title for the second map.

```
title2 "Using only DENSITY values 0 to 2";
```

Show reduced map with density levels 0-2. The DENSITY= option specifies the density levels that are used.

```
proc gmap map=can2
          data=can2 all density=2;
   id province;
   choro province / nolegend coutline=blue;
run;
quit;
```

References

Douglas, D.H. and Peucker, T.K. 1973. “Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature.” *The Canadian Cartographer* 10: 112–122.

Chapter 10

GREMOVE Procedure

Overview: GREMOVE Procedure

The GREMOVE procedure processes a map data set that is used as input. It does not produce any graphics output. Instead, it produces an output data set that typically becomes the input map data set for the GMAP procedure. (See Chapter 7, “GMAP Procedure,” on page 189). The GREMOVE procedure combines unit areas defined in a map data set into larger unit areas by removing shared borders between the original unit areas. For example, Figure 10.1 on page 402 and Figure 10.2 on page 402 show combined unit areas in a typical map data set by removing state boundaries to create regional census divisions.

Figure 10.1 *Map Before Removing Borders*

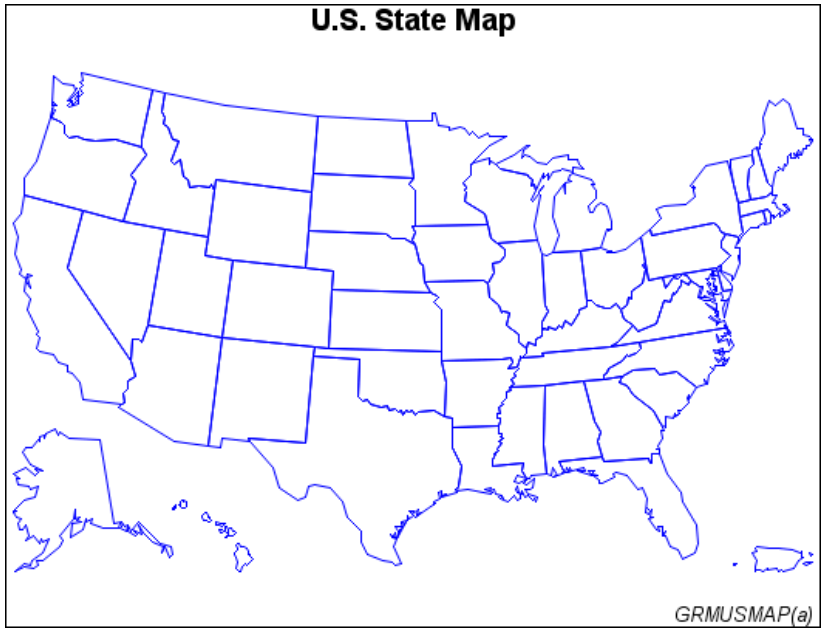
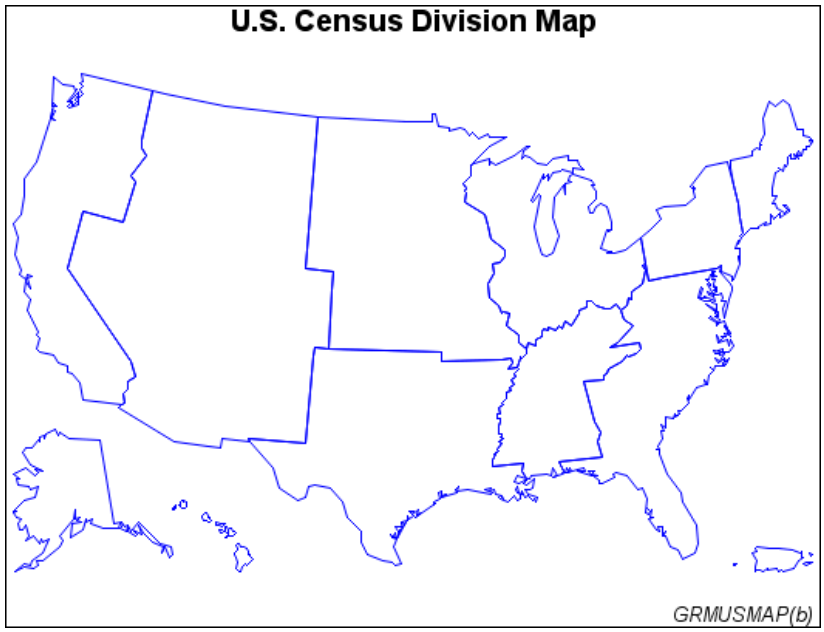


Figure 10.2 *Map After Removing Borders*



The program for these maps is shown in "Example 1: Removing State Boundaries from U.S. Map" on page 408.

About the GREMOVE Procedure

The GREMOVE procedure processes the input map data set to remove internal boundaries and creates a new map data set. The PROC GREMOVE statement identifies

the input and output map data sets. The ID statement identifies the variable or variables in the input map data set that define the current unit areas. The BY statement identifies the variable or variables in the input map data set that define the new unit areas.

About the Input Map Data Set

The input map data set must be in a SAS data set format (see "About Map Data Sets" on page 193) and it must contain these variables:

- a numeric variable named X that contains the horizontal coordinates of the map boundary points.
- a numeric variable named Y that contains the vertical coordinates of the map boundary points.
- one or more variables that uniquely identify the current unit areas in the map. These variables are listed in the ID statement. Each group of observations with a different ID variable value is evaluated as a separate unit area.
- one or more variables that identify the new unit areas to be created in the output map data set. These variables are listed in the BY statement.

It might also contain the variable SEGMENT, which is used to distinguish non-conterminous segments of the same unit areas. Other variables might exist in the input map data set and by default they are carried into the output map data set. The other variables do not affect the GREMOVE procedure, but they can be used by other mapping procedures. If these other variables are not needed, use the DROPVARS option to override this default inclusion behavior and omit them from the output map data set.

About the Output Map Data Set

The output map data set contains the newly defined unit areas. These new unit areas are created by removing all interior line segments from the original unit areas. All variables in the input map data set including X, Y, SEGMENT, and the variables listed in the BY statement are carried into the output map data set by default. Use the DROPVARS option to omit the variables from the output map data set.

The output map data set might contain missing X, Y coordinates that are necessary to construct any polygons with enclosed boundaries. This includes lakes or combined regions that have one or more hollow interior regions.

The SEGMENT variable in the output map data set is ordered according to the size of the bounding box around the polygon that it describes. A SEGMENT value of 1 describes the polygon whose bounding box is the largest in the unit area, and each additional SEGMENT value describes a smaller polygon. This information is useful for removing small polygons that clutter up maps.

All current unit areas with common BY-variable values are combined into a single unit area in the output map data set. The new unit area contains the following:

- all boundaries that are not shared, such as islands and lakes
- all boundaries that are shared by two different BY groups

All other variables can be used by other mapping procedures.

About Unmatched Area Boundaries

If you are using map data sets in which area boundaries do not match precisely (for example, if the boundaries were digitized with a different set of points), PROC GREMOVE is unable to identify common boundaries properly. This results in abnormalities in your output data set.

If the points in the area boundaries match up except for precision differences, round each X and Y value in your map data set accordingly. To accomplish this use the DATA step function ROUND before using PROC GREMOVE. See *SAS Functions and CALL Routines: Reference* for information about the ROUND function.

Take, for example, the map data set APPROX, which has horizontal and vertical coordinate values for interior boundaries of unit areas that are exactly equal only to three decimal places. The following DATA step creates a new map data set, EXACT, that is better suited for use with the GREMOVE procedure:

```
data exact;
   set approx;
   if x ne . then x=round(x,.001);
   if y ne . then y=round(y,.001);
run;
```

You can also use the FUZZ option to specify a level of tolerance so that the boundaries do not need to match precisely.

Syntax: GREMOVE Procedure

Requirements: An input map data set is required. If not specified, an input map data set must have been created before issuing the PROC GREMOVE statement.

The BY and ID statements are required.

PROC GREMOVE DATA=*input-map-data-set*
<DROPVARS>
<FUZZ=*fuzz-factor*>
<OUT=*output-map-data-set*>
<NODECYCLE>;

BY *variable(s)*
<DESCENDING> *variable-1* <<DESCENDING> *variable-2* …>
<NOTSORTED>;

ID *variable(s)*;

PROC GREMOVE Statement

Identifies the input and output map data sets.

Requirement: An input map data set is required. If not specified, an input map data set must have been created before issuing the PROC GREMOVE statement.

Syntax

PROC GREMOVE DATA=*input-map-data-set*
<DROPVARS>
<FUZZ=*fuzz-factor*>
<OUT=*output-map-data-set*>
<NODECYCLE>;

Summary of Optional Arguments

DROPVARS
: requests the removal of most of the variables from the input map data set.

FUZZ=*fuzz-factor*
: specifies a tolerance for possible error in the data.

NODECYCLE | NC
: tells PROC GREMOVE to use a topological algorithm for closing the resulting polygons.

OUT=*output-data-set*
: names the new map data set, which contains the coordinates of the new unit areas created by the GREMOVE procedure.

Required Argument

DATA=*input-map-data-set*
: specifies the map data set that is to be processed. The GREMOVE procedure expects the observations in the input map data set to be sorted in ascending order of the BY-variable values.

Default	The GREMOVE procedure uses the most recently created SAS data set.
Requirement	The DATA= statement is required if no input map data set was created before issuing the PROC GREMOVE statement.
See	"About the Input Map Data Set" on page 403
	"The SAS Data Set: Your Key to the SAS System" in *Step-by-Step Programming with Base SAS*
	"About Data Set Options" in *SAS Data Set Options: Reference*
Example	"Example 2: Creating an Outline Map of Africa" on page 413

Optional Arguments

DROPVARS
: requests the removal of most of the variables from the input map data set. The only variables that are not removed are the variables specified in the ID statement and the X, Y, and SEGMENT variables. It also keeps any variables specified by the DROP and KEEP statements on the DATA step.

Use the DROP statement on the DATA step instead of specifying the DROPVARS option to selectively DROP certain variables. Conversely, indicate any variables that you want to retain by using the KEEP statement on the DATA step along with the DROPVARS option in the PROC GREMOVE statement. Do not use both DROP and KEEP statements within the same DATA step.

Alias DROPV

Default Variable values are kept unless option DROPVARS is specified.

FUZZ=*fuzz-factor*
specifies a tolerance for possible error in the data. This allows for points that are very close but not quite equal to be considered as the same point. The fuzz-factor can be any nonnegative number. A fuzz-factor of 0.0 would indicate that the points have to be exactly the same. The unit represented by the fuzz-factor (degrees, radians, feet, meters, kilometers, miles) is the same as that represented by the X and Y values of the points.

The error is computed the same in both X and Y directions using the following formula: `Point_is_equal = (ABS(x1 - x2) <= fuzz-factor) && (ABS(y1 - y2) <= fuzz-factor)`

NODECYCLE | NC
tells PROC GREMOVE to use a topological algorithm for closing the resulting polygons. By default, PROC GREMOVE simply removes internal boundaries without using any polygon information. This might cause errors in closing the resulting polygons in certain cases—specifically when two resulting polygons intersect at a single point. Using a topological algorithm allows PROC GREMOVE to traverse the resulting polygons for proper closure of the polygons. When the single point intersection is encountered, the algorithm uses the topology to correctly interpret which existing segment to choose in closing the polygon. Therefore, the use of NODECYCLE requires that the data be topologically correct. In other words, polygons do not overlap themselves or each other and there are no anomalies in the boundaries such as a repeated series of points.

Certain SAS/GRAPH procedures, such as PROC GREDUCE, which have no knowledge of topology and do not maintain topology, can produce topologically incorrect polygons. Therefore, it is recommended that you not use PROC GREDUCE if you are going to use PROC GREMOVE with NODECYCLE specified.

OUT=*output-data-set*
names the new map data set, which contains the coordinates of the new unit areas created by the GREMOVE procedure. By default, the GREMOVE procedure names the new data set using the DATA*n* naming convention. That is, the procedure uses the name WORK.DATA*n*, where *n* is the next unused number in sequence. Thus, the first automatically named data set is DATA1, the second is DATA2, and so on.

See "About the Output Map Data Set" on page 403.

Example "Example 2: Creating an Outline Map of Africa" on page 413

BY Statement

Lists the variable or variables that identify the new unit areas.

Requirement: At least one variable is required.

See: "BY Statement" on page 19.

Example: "Example 1: Removing State Boundaries from U.S. Map" on page 408

Syntax

BY *variable(s)*
<DESCENDING> *variable-1* <<DESCENDING> *variable-2 …*>
<NOTSORTED>;

Summary of Optional Arguments

DESCENDING
indicates that the input map data set is sorted in descending order.

NOTSORTED
indicates that observations with the same BY-variable values are to be grouped as they are encountered without regard for whether the values are in alphabetical or numerical order.

Required Argument

variable(s)
identifies one or more variables in the input map data set that define the new unit areas. *Variable(s)* can be either numeric or character.

The BY variables in the input map data set become the ID variables for the output map data set.

Optional Arguments

DESCENDING
indicates that the input map data set is sorted in descending order. By default, the GREMOVE procedure expects all BY-variable values to appear in ascending order.

This option affects only the variable that immediately follows the option.

NOTSORTED
indicates that observations with the same BY-variable values are to be grouped as they are encountered without regard for whether the values are in alphabetical or numerical order. NOTSORTED can appear anywhere in the BY statement. It affects all of the variables that are specified in the statement. NOTSORTED overrides DESCENDING if both appear in the same BY statement.

Details

Ordering Observations

To sort the input map data set, use the SORT procedure in Base SAS, for example:

```
/* arrange the observations in desired order */
proc sort data=mapdata out=mapsort;
   by state;
run;
   /* remove the county boundaries              */
proc gremove data=mapsort out=newmap;
   by state;
   id county;
run;
```

Notice that the GREMOVE procedure uses the same BY statement as the SORT procedure.

See the *Base SAS Procedures Guide* for more information about the SORT procedure.

Note: If an observation is encountered for which the BY-variable value is out of the proper order, the GREMOVE procedure stops and issues an error message.

ID Statement

Identifies the variable or variables that define the hierarchy of the current unit areas in the input map data set.

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

See: "Example 1: Removing State Boundaries from U.S. Map" on page 408

Syntax

ID *id-variable(s)*;

Required Argument

id-variable(s)
specifies one or more variables in the input map data set that identify the unit areas to be combined. These variables are not included in the output map data set. *Id-variable(s)* can be either numeric or character.

See "About the Input Map Data Set" on page 403

Examples: GREMOVE Procedure

Example 1: Removing State Boundaries from U.S. Map

Features: BY statement
ID statement

Other features: SORT procedure, MERGE statement, and LIBNAME statement

Sample library member: GRMUSMAP

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

This example processes the MAPS.US map data set, supplied with SAS/GRAPH, to produce a new map data set containing boundaries for the U.S. Bureau of the Census divisions. Because the MAPS.US map data set does not contain a variable to identify any unit area other than states, this example creates a map data set that contains the census divisions. That map data set can then be processed with the GREMOVE procedure.

The STATE variable in the MAPS.US data set, containing numeric FIPS codes for each state, is used as the BY-variable to merge the CBSTATES and MAPS.US data sets. Output 10.1 on page 409 shows some of the variables that are present in the data set before using the GREMOVE procedure:

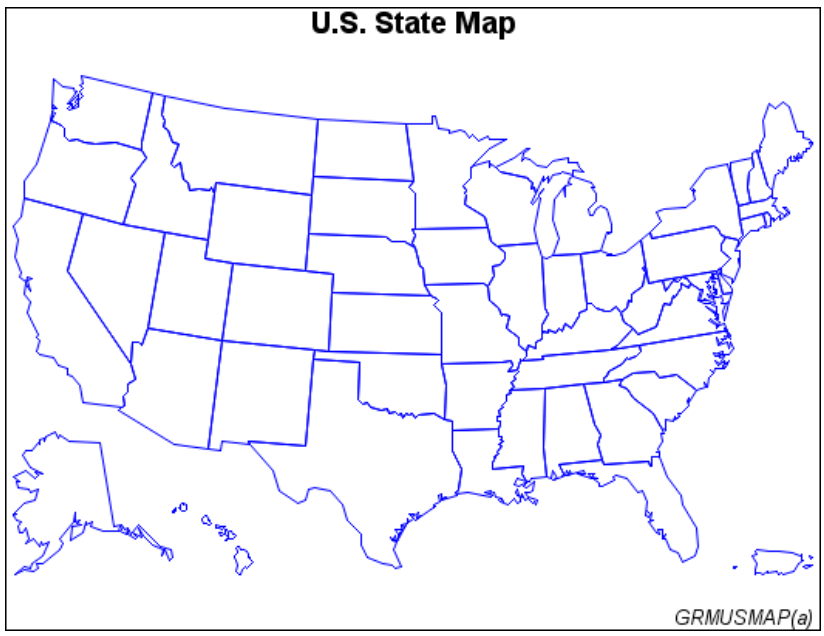
Output 10.1 *The MAPS.US Data Set*

MAPS.US Data Set

OBS	STATE	SEGMENT	X	Y
1	1	1	0.16175	-0.10044
2	1	1	0.12305	-0.10415
3	1	1	0.12296	-0.10678
.				
.				
.				
1524	56	1	-0.18757	0.15035
1525	56	1	-0.10158	0.13997
1526	56	1	-0.10398	0.11343

And Output 10.2 on page 409 shows the map before processing:

Output 10.2 *Map Before Removing Borders*



Output 10.3 on page 410 shows the variables that are present in the data set after you use the GREMOVE procedure. Notice that the new map data set contains a new variable called DIVISION:

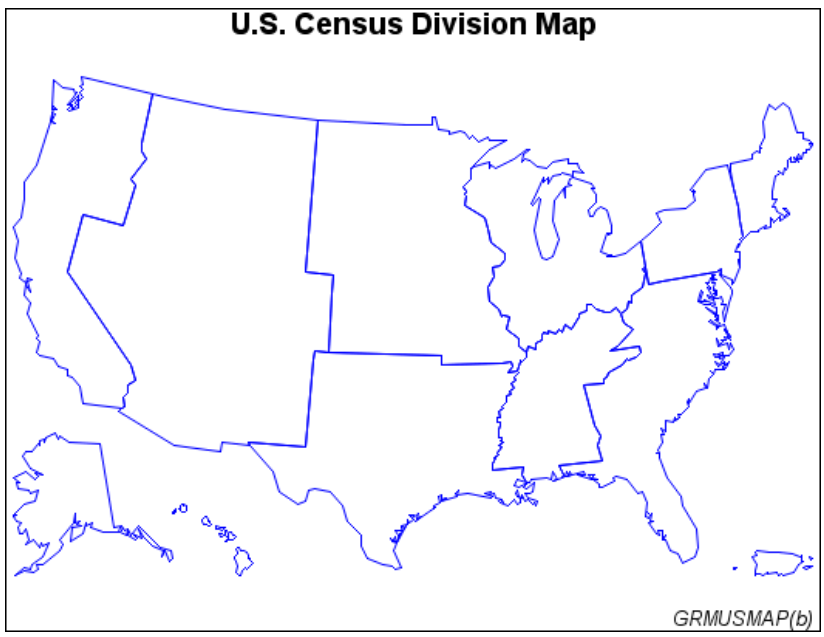
Output 10.3 *The REMSTATE Data Set*

REMSTATE Data Set

OBS	X	Y	SEGMENT	DIVISION
1	0.29825	0.17418	1	1
2	0.29814	0.17820	1	1
3	0.30206	0.18045	1	1
.				
.				
.				
1082	-0.18715	-0.16010	8	9
1083	-0.18747	-0.15971	8	9
1084	-0.18747	-0.15951	8	9

Output 10.4 on page 410 shows the new map after PROC GREMOVE has removed interior state boundaries.

Output 10.4 *Map After Removing Borders*



Program

```
goptions reset=all border;

data cbstates;
   length state 8 stcode $ 2 division 4;
   input stcode division @@;
   state=stfips(stcode);
   drop stcode;
   datalines;
CT 1 MA 1 ME 1 NH 1 RI 1 VT 1 PA 2 NJ 2 NY 2 IL 3 IN 3 MI 3 OH 3 WI 3 IA 4 KS 4
MN 4 MO 4 ND 4 NE 4 SD 4 DC 5 DE 5 FL 5 GA 5 MD 5 NC 5 PR 5 SC 5 VA 5 WV 5
AL 6 KY 6 MS 6 TN 6 AR 7 LA 7 OK 7 TX 7 AZ 8 CO 8 ID 8 MT 8 NM 8 NV 8 UT 8
WY 8 AK 9 CA 9 HI 9 OR 9 WA 9
;

proc sort data=cbstates out=cbsort;
```

```
   by state;
run;

data uscb;
   merge cbsort maps.us;
   by state;
run;

proc sort data=uscb out=divstate;
   by division;
run;

proc gremove data=divstate out=remstate;
   by division;
   id state;
run;

title "U.S. State Map";
footnote j=r "GRMUSMAP(a) ";

pattern value=mempty color=blue;

proc gmap map=maps.us data=maps.us all;
   id state;
   choro state / nolegend levels=1;
run;

title "U.S. Census Division Map";
footnote j=r "GRMUSMAP(b) ";

proc gmap map=remstate data=remstate all;
   id division;
   choro division / nolegend levels=1;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create data set CBSTATES. This data set includes a variable, DIVISION, that contains the number of the U.S. Bureau of the Census division for the state. This DATA step converts letter codes to numeric FIPS codes that match those in the STATE variable of MAPS.US.

```
data cbstates;
   length state 8 stcode $ 2 division 4;
   input stcode division @@;
   state=stfips(stcode);
   drop stcode;
   datalines;
CT 1 MA 1 ME 1 NH 1 RI 1 VT 1 PA 2 NJ 2 NY 2 IL 3 IN 3 MI 3 OH 3 WI 3 IA 4 KS 4
MN 4 MO 4 ND 4 NE 4 SD 4 DC 5 DE 5 FL 5 GA 5 MD 5 NC 5 PR 5 SC 5 VA 5 WV 5
AL 6 KY 6 MS 6 TN 6 AR 7 LA 7 OK 7 TX 7 AZ 8 CO 8 ID 8 MT 8 NM 8 NV 8 UT 8
WY 8 AK 9 CA 9 HI 9 OR 9 WA 9
;
```

Sort data set in FIPS-code order. Create a sorted data set, CBSORT. It can be properly match-merged with the MAPS.US map data set, which is already sorted in FIPS-code order.

```
proc sort data=cbstates out=cbsort;
   by state;
run;
```

Add DIVISION variable to map data set by merging the CBSORT data set with MAPS.US. Create a new map data set, USCB, that contains all of the state boundary coordinates from the MAPS.US data set plus the added variable DIVISION.

```
data uscb;
   merge cbsort maps.us;
   by state;
run;
```

Sort data set in DIVISION order. Sort USCB by the DIVISION variable to create the DIVSTATE data set.

```
proc sort data=uscb out=divstate;
   by division;
run;
```

Remove interior boundaries within divisions. BY specifies the variable, DIVISION, in the input map data set that identifies the new unit areas. ID specifies the variable, STATE, in the input map data set that identifies the current unit areas.

```
proc gremove data=divstate out=remstate;
   by division;
   id state;
run;
```

Define title and footnote for map.

```
title "U.S. State Map";
footnote j=r "GRMUSMAP(a) ";
```

Define pattern characteristics.

```
pattern value=mempty color=blue;
```

Show the original map.

```
proc gmap map=maps.us data=maps.us all;
   id state;
   choro state / nolegend levels=1;
run;
```

Define new title and footnote for map.

```
title "U.S. Census Division Map";
footnote j=r "GRMUSMAP(b) ";
```

Show the regional map. ID specifies the variable, DIVISION, that identifies the unit areas in the processed data set. CHORO specifies DIVISION as the response variable.

```
proc gmap map=remstate data=remstate all;
   id division;
```

```
   choro division / nolegend levels=1;
run;
quit;
```

Example 2: Creating an Outline Map of Africa

Features: PROC GREMOVE options DATA= and OUT=

Other features: GMAP procedure

Sample library member: GRMAFRIC

Notes: The SAS Sample Library is not available in SAS Studio.You can download the SAS/GRAPH samples in the SAS Sample Library in zipped form from the SAS/GRAPH product documentation page on support.sas.com.

The SAS-supplied map data set(s) used in this program might not be an available resource on your system.

This example processes the MAPS.AFRICA map data set, supplied with SAS/GRAPH, to produce a new map data set that contains no internal boundaries. This is done by adding a new variable, REGION, to the map data set and setting it equal to 1. Unit areas from the input map data set that have the same BY-variable value are combined into one unit area in the output map data set.

Output 10.5 on page 413 shows some of the variables that are present in the original map data set:

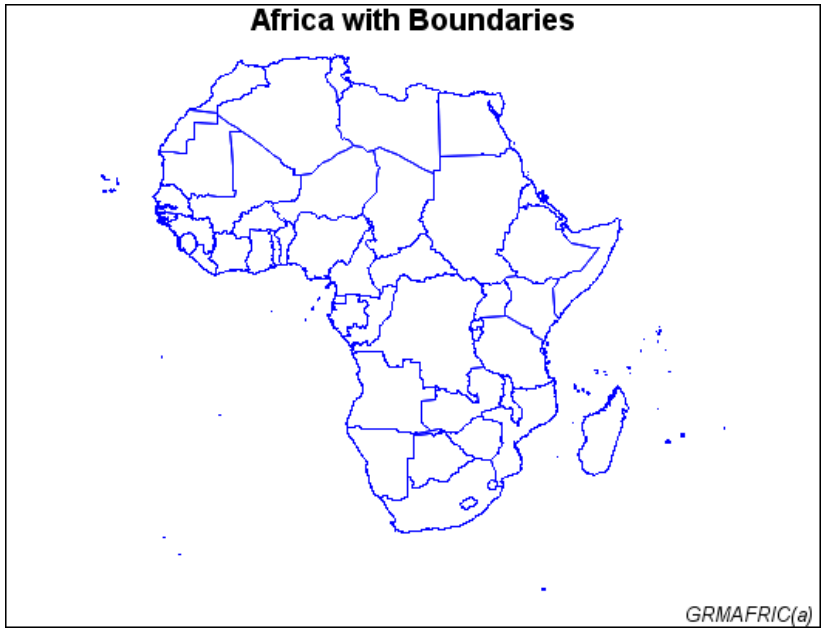
Output 10.5 *The MAPS.AFRICA Data Set*

MAPS.AFRICA Data Set

OBS	ID	SEGMENT	X	Y
1	125	1	0.57679	1.43730
2	125	1	0.57668	1.43467
3	125	1	0.58515	1.42363
.				
.				
.				
3462	990	1	1.04249	0.50398
3463	990	1	1.04184	0.50713
3464	990	1	1.04286	0.50841

Output 10.6 on page 414 shows the map before processing:

Output 10.6 *Map Before Removing Borders (GRMAFRIC(a))*



The new AFRICA map data set is created with a new variable, REGION. Output 10.7 on page 414 shows the variables that are present in the new map data set created by the GREMOVE procedure:

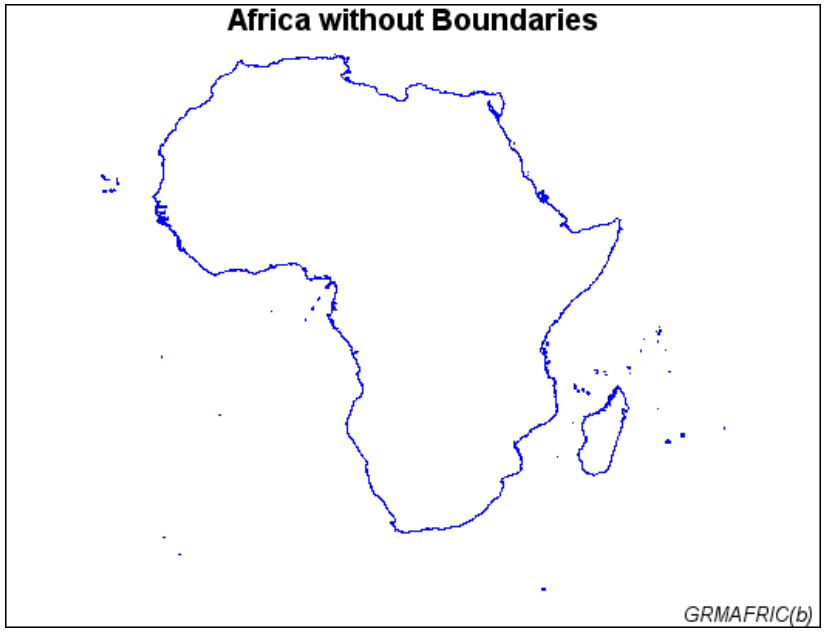
Output 10.7 *The AFRICA Data Set*

AFRICA Data Set

OBS	X	Y	SEGMENT	REGION
1	0.24826	1.02167	1	1
2	0.25707	1.02714	1	1
3	0.26553	1.03752	1	1
.				
.				
.				
982	1.19071	1.30043	3	1
983	1.18675	1.30842	3	1
984	1.18518	1.32822	3	1

Output 10.8 on page 415 shows the new map after PROC GREMOVE has removed all of the interior boundaries:

Output 10.8 *Map After Removing Borders*



Program

```
goptions reset=all border;

data newaf;
   set maps.africa;
   region=1;
run;

proc gremove data=newaf out=africa;
   by region;
   id id;
run;

title "Africa with Boundaries";
footnote j=r "GRMAFRIC(a) ";

pattern value=mempty color=blue;

proc gmap map=maps.africa data=maps.africa all;
   id id;
   choro id / nolegend levels=1;
run;

title "Africa without Boundaries";
footnote j=r "GRMAFRIC(b) ";

proc gmap data=africa map=africa;
   id region;
   choro region / nolegend levels=1;
run;
quit;
```

Program Description

Set the graphics environment.

```
goptions reset=all border;
```

Create the NEWAF data set. This new map data set contains all the variables in the MAPS.AFRICA map data set supplied with SAS/GRAPH plus the added variable REGION.

```
data newaf;
   set maps.africa;
   region=1;
run;
```

Remove the unit areas from the AFRICA data set. DATA= specifies the input map data set and OUT= specifies the output map data set. The input map data set has a variable called REGION that is used as the BY-variable to identify the new unit areas. The ID statement specifies the current unit areas from the input map data set.

```
proc gremove data=newaf out=africa;
   by region;
   id id;
run;
```

Define the title and footnote.

```
title "Africa with Boundaries";
footnote j=r "GRMAFRIC(a) ";
```

Define pattern characteristics.

```
pattern value=mempty color=blue;
```

Display the original map.

```
proc gmap map=maps.africa data=maps.africa all;
   id id;
   choro id / nolegend levels=1;
run;
```

Define a new title and footnote for the map.

```
title "Africa without Boundaries";
footnote j=r "GRMAFRIC(b) ";
```

Display the map with no boundaries. ID specifies the variable, REGION, that identifies the unit areas in the processed data set.

```
proc gmap data=africa map=africa;
   id region;
   choro region / nolegend levels=1;
run;
quit;
```

Chapter 11

MAPIMPORT Procedure

Overview: MAPIMPORT Procedure

The MAPIMPORT procedure enables you to import Esri shapefiles (spatial data formats) and process the SHP files into map data sets that are made available with SAS/GRAPH. See "About Traditional Map Data Sets" on page 194 for more information.

The MAPIMPORT procedure does not produce any graphics output. Instead, it produces an output map data set, which can be used with the GMAP procedure or other mapping procedures.

The shapefiles file types are described in the following table:

Table 11.1 *Shapefiles File Types*

File Extension	Description
.dbf	Identification information (field-identifier names and values) assigned to specific polygons
.shx	Shape information for the polygons that compose the map. [1]

[1] These files are used with SHP files and cannot be imported by themselves.

File Extension	Description
.shp	Combines the shape information for the polygons that compose the map and the identification information (field-identifier names and values) assigned to the specific polygons

Note: If you import a very highly detailed map, then the GMAP procedure might produce extraneous lines when drawing it. To avoid this issue, use the GREDUCE procedure to reduce the number of map points.

Syntax: MAPIMPORT Procedure

Requirement: The name and location of an output data set and the complete path for the input data file.

Note: The single quotation marks surrounding field identifiers are optional when the field identifiers follow the SAS naming convention. Single quotation marks are required for field identifiers that are nonstandard SAS names. When field identifiers placed in single quotation marks are nonstandard SAS names, the field identifiers are converted to a standard SAS name in the traditional map data set. For more information about the standard SAS naming convention, see "Names in the SAS Language" in *SAS Language Reference: Concepts* in the *SAS Language Reference: Concepts* . For more information about how invalid field identifiers placed in single quotation marks are renamed, see "VALIDVARNAME= SAS System Option" in *SAS/ACCESS for Relational Databases: Reference* in the *SAS/ACCESS for Relational Databases: Reference*.

PROC MAPIMPORT OUT=*map-data-set* DATAFILE= *'path-to-shapefile'*
<CONTENTS>
<CREATE_ID_>;
  EXCLUDE '*field-identifier(s)*';
  ID '*field-identifier(s)*';
  RENAME '*field-identifier-1*'=*variable-name-1* < '*field-identifier-2*'=*variable-name-2* …>;
  SELECT '*field-identifier(s)*';

PROC MAPIMPORT Statement

Identifies the input Esri shapefile and converts this map into a SAS/GRAPH map data set.

Requirement: The name and location of an output data set and the complete path for the input data file.

Syntax

PROC MAPIMPORT OUT=*map-data-set* DATAFILE='*path-to-shapefile* '
<CONTENTS>
<CREATE_ID_>;

Summary of Optional Arguments

CONTENTS
: displays information about the shapefile, including field identifier names and types.

CREATE_ID_
: creates a map ID variable named _ID_ with a unique value for each polygon in the map.

Required Arguments

OUT=*map-data-set*
: specifies the name of the output map data set that is created.

DATAFILE='*path-to-shapefile*'
: specifies the path and filename of the shapefile that is read and processed.

Note: By default, all of the fields in a shapefile are included in the output map data set. To include only specific fields in the output map data set, use the SELECT statement. To exclude specific fields from the output map data set, use the EXCLUDE statement.

Alias INFILE=

Optional Arguments

CONTENTS
: displays information about the shapefile, including field identifier names and types.

CREATE_ID_
: creates a map ID variable named _ID_ with a unique value for each polygon in the map. This variable is created automatically if the DBF file is missing.

Interaction This statement has no effect if you also specify the ID statement.

EXCLUDE Statement

Specifies one or more fields from the shapefile that are excluded from the output map data set.

Restrictions: If you specify conflicting values for the EXCLUDE and SELECT statements, then the MAPIMPORT procedure produces an error.

If you specify the same field identifier in the EXCLUDE statement and in the ID statement, then the MAPIMPORT procedure produces an error.

Requirement: At least one *field-identifier* is required.

Syntax

EXCLUDE '*field-identifier(s)*';

Required Argument

'*field-identifier(s)*'

specifies one or more fields from the shapefile that are excluded from the output map data set. All of the fields that you do not specify are included in the output map data set.

If you do not specify the EXCLUDE statement or the SELECT statement, then all of the fields from the shapefile are included in the output map data set.

ID Statement

Reorders the map polygons by one or more identifier fields.

Requirement: At least one *field-identifier* is required.

Interaction: The CREATE_ID option in the PROC MAPIMPORT statement has no effect when you also specify the ID option.

Syntax

ID '*field-identifier(s)*';

Required Argument

'*field-identifier(s)*'

specifies one or more fields in the shape file that identify the polygons in the map. The values of the fields that you specify are used to reorder the map polygons and assign segment numbers in the output map data set.

When you do not specify the ID statement, the MAPIMPORT procedure uses the existing polygon order for the output map data set.

You might want to use the ID statement when the default output map data set does not draw properly in the GMAP procedure. If the ID variable that you specify in the GMAP procedure is not unique for each polygon, then extraneous lines might appear in your GMAP output. To ensure that the ID variable is unique for each polygon, specify the same ID statement in both the MAPIMPORT and GMAP procedures.

RENAME Statement

Renames variables in the output map data set that correspond to specific fields in the shapefile.

Requirement: At least one *field-identifier* and *variable-name* pair are required.

Syntax

RENAME '*field-identifier-1*'=*variable-name-1* < '*field-identifier-2*'=*variable-name-2* … >;

Required Argument

'field-identifier'=variable-name

assigns a variable name in the output map data set for a field in the shapefile. You can specify multiple field identifier and variable name pairs, separated by a space.

For example, the following code renames the STNAME field to STATE, and the FIPSTATE field to STATE_FIPS: **`rename "stname" = state "fipstate" = state_fips;`**

By default, when you do not specify the RENAME statement, the MAPIMPORT procedure uses the field name in the shapefile as the variable name in the output map data set. However, if the field name is not a valid SAS variable name, then the variable name is modified in the output map data set. For more information about valid SAS variable names, see "Names in the SAS Language" in *SAS Language Reference: Concepts* in the *SAS Language Reference: Concepts*.

SELECT Statement

Selects the fields from the shapefile that are included in the output map data set.

Restriction: If you specify conflicting values for the EXCLUDE and SELECT statements, then the MAPIMPORT procedure produces an error.

Requirement: At least one *field-identifier* is required.

Syntax

SELECT '*field-identifier(s)*';

Required Argument

'field-identifier(s)'

specifies one or more fields from the shapefile that are included in the output map data set. If you do not use the SELECT statement or the EXCLUDE statement, then all of the fields from the shapefile are included in the output map data set.

For field identifiers that are not valid SAS variable names, the MAPIMPORT procedure changes the name of the variable in the output map data set automatically. For more information about valid SAS variable names, see "Names in the SAS Language" in *SAS Language Reference: Concepts* in the *SAS Language Reference: Concepts*.

Examples: MAPIMPORT Procedure

Example 1: Including All Variables from the SHP Shapefile

In the following example, World30.shp contains polygons that compose a political boundary world map. All the field identifiers in the World30.shp file are included in the traditional map data set, MYWORLD.

This example and the remaining examples in this section use shapefiles with the .shp and .dbf extensions. Specify your shapefiles locations and replace the example filenames and field identifiers with information from your shapefiles in order to run these examples.

```
PROC MAPIMPORT OUT=myworld
DATAFILE="pathname-of-shapefile-location
'\' or '/'world30.shp";
run;
```

Example 2: Including Selected Variables from the SHP Shapefile

In the following example, the STATES.SHP file contains polygons that compose the political boundaries of a U.S. states map. Only the STATE_FIPS (the state FIPS codes), STATE_NAME (the state name), and STATE_ABBR (the two letter state abbreviation) variables are included in the traditional map data set, MYSTATES. STATE_FIPS is renamed FIPS, STATE_NAME is renamed STATE, and STATE_ABBR is renamed ABBREV in the MYSTATES map data set.

```
PROC MAPIMPORT OUT=mystates
DATAFILE="pathname-of-shapefile-location
'\' or '/'states.shp";
     SELECT STATE_FIPS STATE_NAME STATE_ABBR;
     RENAME STATE_FIPS=FIPS STATE_NAME=STATE STATE_ABBR=ABBREV;
run;
```

Example 3: Excluding a Variable from the SHP Shapefile

In the following example, the STATES.SHP file contains polygons that compose the political boundaries of a U.S. state map. The variable OTHER is excluded from the traditional map data set, MYSTATES2.

```
PROC MAPIMPORT OUT=mystates2
DATAFILE="pathname-of-shapefile-location
'\' or '/'states.shp";
     EXCLUDE OTHER;
run;
```

Example 4: Using the ID Statement

In the following example, the shapefile is a ZCTA file from the US Census Bureau that contain polygons that are based on ZIP codes. The ZCTA field is the identifier that you want to use, but the polygons in the shapefile do not have unique values for ZCTA. If you do not specify the ID statement, then the GMAP procedure draws extra lines between the map areas for ZCTA.

Identify the ZCTA field in the ID statement. This ensures that the polygons for each value of ZCTA are grouped together and assigned different SEGMENT values in the output map data set. The GMAP procedure can now draw the map areas for ZCTA correctly.

```
PROC MAPIMPORT OUT=myzcta
DATAFILE="pathname-of-shapefile-location
'\' or '/'zt06_d00.shp";
```

```
  ID zcta;
run;
```

Example 5: Including Selected Variables from the DBF Shapefile

In the following example, the STATES.DBF file contains the identification information (field-identifier names and values) applied to the U.S. states polygon map. Only the STATE_FIPS (the state FIPS codes), STATE_NAME (the state names), and STATE_ABBR (the two letter state abbreviations) variables are included in the traditional map data set, MYDATA. STATE_FIPS is renamed FIPS, STATE_NAME is renamed STATE, and STATE_ABBR is renamed ABBREV in the MYDATA map data set.

```
PROC MAPIMPORT OUT=mydata
DATAFILE="pathname-of-shapefile-location
'\' or '/'states.dbf";
     SELECT STATE_FIPS STATE_NAME STATE_ABBR;
     RENAME STATE_FIPS=FIPS STATE_NAME=STATE STATE_ABBR=ABBREV;
run;
```

Recommended Reading

Here is the recommended reading list for this title:

- *SAS/GRAPH: Beyond the Basics*
- *SAS/GRAPH: Network Visualization Workshop User's Guide*
- *SAS/GRAPH: Reference*
- *Base SAS Procedures Guide*
- *Base SAS Utilities: Reference*
- *Maps Made Easy Using SAS*
- *Output Delivery System: The Basics and Beyond*
- *The How-to Book for SAS/GRAPH Software*
- *SAS Data Set Options: Reference*
- *SAS Formats and Informats: Reference*
- *SAS Functions and CALL Routines: Reference*
- *SAS Language Reference: Concepts*
- *SAS Statements: Reference*
- *SAS System Options: Reference*
- *Statistical Graphics Procedures by Example: Effective Graphs Using SAS*
- *Statistical Graphics in SAS: An Introduction to the Graph Template Language and the Statistical Graphics Procedures*
- SAS offers instructor-led training and self-paced e-learning courses to help you get started with SAS/GRAPH. For more information about the courses available, see sas.com/training.

For a complete list of SAS publications, go to sas.com/store/books. If you have questions about which titles you need, please contact a SAS Representative:

SAS Books
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-0025
Fax: 1-919-677-4444
Email: sasbook@sas.com
Web address: sas.com/store/books

Glossary

absolute coordinate
a coordinate that is measured from the origin of a coordinate system.

ActiveX
a technology developed by Microsoft that is used to add interactivity to web pages.

anti-aliasing
a rendering technique for improving the appearance of text and curved lines in a graph by blurring the jagged edges normally present. The degree of improvement is relative to the nature of the graphical content (for example, vertical and horizontal lines do not benefit from anti-aliasing). Extra processing is required to perform anti-aliasing.

area bar chart
a bar chart that applies an additional magnitude of width to the bars that results in categorized bars. Each bar has both a height and a width measure that can be independent of each other.

aspect ratio
the ratio of a shape's width to its height in an output area such as a display, plotter, or film recorder.

attribute bundle
a common collection of visual properties associated with a graphical primitive such as a line, marker, or text. For example, all lines have visual properties of pattern, thickness, and color. All markers have visual properties of symbol, size, weight, and color. Attribute bundles can be associated with style elements in order to indirectly assign visual properties.

axis
a line that represents the midpoints (for a discrete axis) or the scale (for a continuous or interval axis) for graphing variable or data values. An axis typically consists of an axis line with tick marks, tick values (or midpoint values), and a label. *See also* major axis, minor axis, Cartesian coordinate system.

axis area
an area bounded by axes, which might be enclosed by an axis line.

baseline
in a font, the imaginary line upon which the characters rest.

block map
a three-dimensional map that uses blocks of varying heights to represent the value of a variable for each map area.

boundary
in the GMAP procedure, a separating line or point that distinguishes between two or more unit areas or segments.

bullet chart
a KPI chart consisting of a bar divided into qualitative ranges according to spcified boundary values. The actual value of the KPI is indicated with a black line, or bullet, down the center of the graph. The target value, if it is specified, is displayed as a vertical line (in a horizontal bullet graph) or a horizontal line (in a vertical bullet graph) across the graph.

BY group
a group of observations or rows that have the same value for a variable that is specified in a BY statement. If more than one variable is specified in a BY statement, then the BY group is a group of observations that have a unique combination of values for those variables.

BY-group processing
the process of using the BY statement to process observations that are ordered, grouped, or indexed according to the values of one or more variables. Many SAS procedures and the DATA step support BY-group processing. For example, you can use BY-group processing with the PRINT procedure to print separate reports for different groups of observations in a single SAS data set.

capline
the highest point of a normal uppercase letter. In some fonts, the capline might be above the top of the letter to allow room for an accent.

Cartesian coordinate system
the two- or three-dimensional coordinate system in which perpendicular axes meet at the origin (0,0) or (0,0,0). Typically, Cartesian coordinate axes are called X, Y, and Z.

catalog
See SAS catalog.

catalog entry
See SAS catalog entry.

category variable
a classification variable with a finite number of distinct (discrete) values. These variables are typically used to split data into subsets. For example, in a bar chart, each unique value is displayed as a bar on a DISCRETE axis. In another example, the variable payment mode can have two values, prepaid and postpaid. Customers can be classified based on this variable as prepaid customers and postpaid customers.

cell
See character cell.

cell header
a graphical element (typically rendered as text or a legend) that is aligned at the top of a cell and provides information about the cell contents. A cell header is defined within a cell block, which is available only within a LATTICE layout.

center point
the location in the GRAPH window that, in conjunction with a radius point, defines the placement and shape of an ellipse or a pie.

CGM
See computer graphics metafile.

character cell (cell)
in device-based SAS/GRAPH procedures, a unit of measure whose size and shape is determined by both the size of the graphics output area and by the number of rows and columns in the graphics output area.

character up vector
the angle at which a character is positioned. The character up vector has two components, x and y, which determine the angle.

character value
a value that can contain alphabetic characters, the numeric characters 0 through 9, and other special characters.

chart statistic
the statistical value calculated for the chart variable: frequency, cumulative frequency, percentage, cumulative percentage, sum, or mean.

chart variable
a variable in the input data set whose values are categories of data represented by bars, blocks, slices, or spines.

chart vertex
a point on a radar chart where a statistical value intersects the spokes.

choropleth map
a two-dimensional map that uses color and fill pattern combinations to represent different categories or levels of magnitude.

class variable
See classification variable.

classification level
for a single classification variable, each unique value is regarded as a classification level. For two or more variables, a classification level is one of the unique combinations (crossings) of the unique values of each variable. For example, if three variables have four, two, and three distinct values, there are 24 classification levels.

classification variable (class variable)
a variable whose values are used to classify the observations in a data set into different groups that are meaningful for analysis. A classification variable can have either character or numeric values. Classification variables include group, subgroup, category, and BY variables.

CMYK
a color coding scheme that specifies a color in terms of the levels of cyan, magenta, yellow, and black components. The level of each component ranges from 0 to 255.

color list
the list of foreground colors that are available for graphics output. The color list is either the default list established from the style, the list created from the device entry, or the list established from the colors specified with the COLORS= graphics option. The colors are derived from either the main color scheme models (NAME, RGB, RGBA, HLS), or from the secondary color scheme models (CMYK, GRAY, HSV).

color map
in SAS/GRAPH software, a table that is used to translate the original colors in graphics output to different colors when replaying graphics output using the GREPLAY procedure. The table is contained in a catalog entry.

column axis
an external axis appearing above or below a column of cells and serving as a common reference for the column of a multi-cell layout, such as a LATTICE, DATAPANEL, or DATALATTICE layout.

column gutter
the space between columns of cells in a multi-cell layout.

column header
text that labels the column contents in a multi-cell layout. This text can be aligned above or below the cells in a column. In a LATTICE layout, the column header is not restricted to text (it can contain a plot or a legend, for example).

column major order
an order for populating cells of a layout or entries in a legend when the number of rows is specified. By default, cells or entries are filled starting from the top left and moving down. When the bottom row of the first column is filled, a new column begins filling to the right of the previous column, and so on until all content items have been placed in cells or entries. There might be empty cells or entries in the last column.

column weight
in a LATTICE layout, the proportion of width allotted to a specific column of the layout. The sum of all column weights is 1.

computer graphics metafile (CGM)
a graphics output file written in the internationally recognized format for describing computer graphics images. This standardization allows any image in a CGM to be imported and exported among different systems without error or distortion.

conditional logic
syntax that enables one set of statements or an optional alternate set of statements to execute at run time.

confidence limits
the upper and lower values of a (usually 95%) confidence interval. In repeated sampling, approximately (1-alpha)*100% of the resulting intervals would contain the true value of the parameter that the interval estimates (where alpha is the confidence level associated with the interval).

continuous legend
a legend that shows a mapping between a color ramp or color segments and corresponding numeric values. Plots that support a COLORMODEL= option can use this type of legend.

contour plot
a three-variable plot that uses line styles or patterns to represent levels of magnitude of z corresponding to x and y coordinates.

coordinate
a value that represents the location of a data point or a graphics element with respect to a coordinate system.

coordinate system
the context in which to interpret coordinates. Coordinate systems vary according to their origin, limits, and units.

Critical Success Factor object (CSF object)
a graphical object that shows the relative position of a specific numeric value within a range of values. Many attributes of the object, such as its general shape, the type of arrow indicator, and how or whether the range is segmented, can be customized.

crossing
a combination of the unique values of one or more classification variables.

CSF object
See Critical Success Factor object.

cube
in three-dimensional graphics, the outlines formed by the intersection of three pairs of parallel planes; each pair is orthogonal to the primary X, Y, and Z axes. The display of the cube is optional.

data area
the portion of the graphics output area in which data values are displayed. The data area is bounded by axes or map areas. In the Annotate facility, the data area defines a coordinate system. *See also* graphics output area, procedure output area, coordinate system.

data object
a transient version of a SAS data set created by ODS. When an input SAS data set is bound to a compiled graph template, an ODS data object is created, based on all the columns requested in the template definition and any new columns that have been directly or indirectly computed. A data object can persist when used with the ODS OUTPUT statement.

data set
See SAS data set.

data tip
data or other detailed information that is displayed when a user positions a mouse pointer over an element in a graph. For example, a data tip typically displays the data value that is represented by a bar, a plot point, or some other element.

data value
a unit of character, numeric, or alphanumeric information that is stored as a single item in a data record.

date value
See SAS date value.

density plot
a univariate probability density curve based on one continuous numeric variable containing a random population sample. The curve represents a theoretical probability density function for a large population. Density plots support density functions such as Normal distributions and Kernel Density Estimates.

density value
a value assigned to each observation in a map data set reflecting the amount of detail (resolution) contributed by the observation.

descriptor information
information about the contents and attributes of a SAS data set. For example, the descriptor information includes the data types and lengths of the variables, as well as which engine was used to create the data. SAS creates and maintains descriptor information within every SAS data set.

design size
the intended size of a graph that is specified in the graph template definition. The DESIGNHEIGHT and DESIGNWIDTH options of the BEGINGRAPH statement set the intended height and width, which are used to determine the scale factors when the graph is resized. The intended height and width are used unless overridden by the ODS Graphics statement HEIGHT or WIDTH options when the template is executed.

device driver
in SAS/GRAPH software, a routine that generates the specific machine-language commands needed to display graphics output on a particular device. SAS/GRAPH device drivers take device-independent graphics information produced by SAS/GRAPH procedures and create the commands required to produce the graph on the particular device.

device entry
a SAS catalog entry of type DEV that stores the values of device parameters (or the characteristics) that are used with a particular output device.

device map
a catalog entry used to convert the SAS/GRAPH internal encoding for one or more characters to the device-specific encoding needed to display the characters in hardware text on a particular graphics output device. *See also* hardware character set.

device parameter
a value in a device entry that defines a default behavior or characteristic of a device driver. Some device parameters can be overridden by graphics options. *See also* graphics option.

device-independent catalog entry
a SAS catalog entry that contains graphics output in a generic format (not device-specific). A device-independent catalog entry can be replayed on any device supported by SAS/GRAPH software.

device-resident font
a font stored in an output device.

dial chart
a KPI chart in the form of a dial divided into qualitative ranges according to the boundary values. The actual value of the KPI is indicated with a large, white triangle pointer. The target value, if it is specified, is displayed as a small, black triangle.

discrete axis
an axis for categorical data values. The distance between ticks has no significance. A bar chart always has a discrete axis.

discrete legend
a legend that provides values or descriptive information about graphical elements in a grouped or overlaid plot.

document file
a file output by the Output Delivery System (ODS) that contains an image or is used to view an image. Examples include HTML, PDF, RTF, SVG, and PostScript files.

drill down
to explore data and access information by moving from summary information to more detailed data from which the summary is derived. For example, you could click folders in a hierarchy from the top downwards to find a specific file. Drilling down provides a method of exploring multidimensional data by moving from one level of detail to the next.

drop line
a line drawn from a point in the plot area perpendicular to an axis.

eavesdrop connection
a method of configuring plotters or other graphics hardcopy devices in which the hardcopy device is connected between an asynchronous terminal and a host computer. All transmissions between the terminal and the host go through the hardcopy device, which is said to be eavesdropping. The hardcopy device looks for and intercepts graphics commands from the host.

ellipse plot
a plot that displays a confidence or prediction ellipse for a set of points.

end angle
the measure in degrees from a particular position within a shape to its trailing edge. For a pie, it is measured from the 3:00 position. For an ellipse, it is measured from the major axis.

entry type
a characteristic of a SAS catalog entry that identifies the catalog entry's structure and attributes to SAS. When you create a SAS catalog entry, SAS automatically assigns the entry type as part of the name. *See also* SAS catalog entry.

equated axes
in two-dimensional plots, axes that use the same drawing scale (ratio of display distance to data interval) on both axes. For example, an interval of 2 on the X axis maps to the same display distance as an interval of 2 on the Y axis. The aspect ratio of the plot display equals the aspect ratio of the plot data. In other words, a 45-degree slope in data will be represented by a 45-degree slope in the display. Equated axes

are always of TYPE=LINEAR. The number of intervals displayed on each axis does not have to be the same.

external axis
an axis that is outside all cells of a layout. An external axis represents a common scale for all plots in a row or column of a multi-cell layout.

fill color
the color of a pattern in a filled, closed graphics object, such as a bar segment, a pie slice, or a map area.

fill pattern
a design of parallel or crosshatched lines, solid colors, or empty space used to fill an area in a graph.

fill style
the visual attributes for a part of a graph (for example, the inside of bar in a bar chart, or the background of a scatter plot) that is filled in with the same color or pattern.

film recorder
a device that receives graphics data streams and records them in the form of photographic images (usually 35-millimeter slides).

fit policy
one of several algorithms for avoiding tick-value collision when space allotted to a predefined area does not permit all the text to fit. For example, an axis might have a THIN policy that eliminates the display of tick values for alternate ticks. A ROTATE policy would turn the tick values at a 45-degree angle. A TRUNCATE policy would truncate all long tick values to a fixed length and add an ellipsis (. . .) at the end to imply truncation. A STAGGER policy would create two rows of tick values with consecutive tick values alternating between rows. A compound policy such as STAGGERROTATE could be used to automatically choose the best fit policy for the situation.

font
a typeface with a specific character shape, spacing, weight, and size. The characters in a font can be figures, symbols, or alphanumeric.

font family
a set of one or more typefaces that share common design characteristics such as serifs, proportional or uniform spacing, or special symbols. For example, Helvetica, Arial, and Albany AMT are members of a sans-serif, proportional font family.

font maximum
in the GFONT procedure, the highest vertical coordinate in a font.

font minimum
in the GFONT procedure, the lowest vertical coordinate in a font.

font unit
in the GFONT procedure, a unit within a range that is defined by coordinates specified in the font data set. For example, a font in which the vertical coordinates range from 10 to 100 has 90 font units.

FreeType font-rendering
a method of rendering fonts that uses the FreeType engine to access the content of font files in order to render high-quality fonts for ODS and SAS/GRAPH. The FreeType engine can be used in all SAS operating environments.

frequency variable
in an input data set, a non-negative and non-zero integer variable that represents the frequency of occurrence of the current observation, essentially treating the data set as if each observation appeared n times, where n is the value of the FREQ variable for the observation.

fringe plot
a plot consisting of short, equal-length line segments drawn from and perpendicular to an axis. Each observation of a numeric variable corresponds to the location for a line segment.

function
See SAS function.

geo-variable
in a feature table, the $GEOREF formatted variable that stores the spatial information as a geometry object.

global selection mode
a default feature that applies a data selection to all graphs and data tables that are associated with the project. Global selection mode enables data to be subset graphically, and at a single level. *See also* local selection mode.

global statement
a SAS statement that you can specify anywhere in a SAS program.

glyph
the most basic element (a grapheme or combination of graphemes) of a typeface or font that carries meaning in the text of a writing system. For example, the Z character can be represented by a number of different glyphs--boldface, italic, or in varying font styles, all of which represent the the letter "Z."

graphical area
in SAS/GRAPH output, the area between the title and footnote area in which the graphics visualization is displayed. The graphical area consists of one or more cells.

graphics device
See graphics output device.

graphics element
a discrete visual part of a picture. For example, a bar in a chart and a plot's axis label are both graphics elements.

graphics object
a discrete visual element of a graph or picture (for example, a bar in a chart, a polygon, a plot's axis, and so on).

graphics option
in a SAS GOPTIONS statement, an option that controls some attribute of the graphics output. The specified value remains in effect only for the duration of the

SAS session. Some graphics options override parameters that have been specified for a graphics output device.

graphics output
output from a graphics program that can be stored as a catalog GRSEG entry or as a graphics stream file. *See also* device-independent catalog entry, graphics output device.

graphics output area
the area of a graphics output device where the graphics output is displayed or drawn. Typically, the graphics output area occupies the full drawing area of the device, but the dimensions of the graphics output area can be changed with graphics options or device parameters. *See also* procedure output area, graphics output device.

graphics output device (graphics device, hardcopy device)
any terminal, printer, or other output device that is capable of displaying or producing graphical output.

graphics output file
a file that contains bitmapped or vector graphic information.

graphics primitive
a function that draws a graphics element.

graphics stream file (GSF)
a file that contains device-dependent graphics commands from a SAS/GRAPH device driver. This file can be sent to a graphics device or to other software applications.

graphics template
See ODS template.

gray scale
a color-coding scheme that specifies a color in terms of gray components. Gray-scale color codes are commonly used with some laser printers and PostScript devices.

grid
a uniform arrangement of the rows and columns of a multi-cell layout.

grid point
a grid location in the GRAPH window that is marked by a dot. Grid points are used for precision placement of objects.

grid request
in the G3GRID procedure, the request specified in a GRID statement that identifies the horizontal variables that identify the x, y plane and one or more z variables for the interpolation.

gridded data
input that contains at least three numeric variables. Two of the variables are treated as X and Y variables and the third variable Z is treated as if it were a function of X and Y. The X and Y variable values occur at uniformly spaced intervals (although the size and number of intervals might be different for X and Y). All X,Y pairs are unique, and Z values are interpolated so that every X,Y pair has a Z value. Raw data that has at least three numeric variables can be converted to gridded data with the

G3GRID procedure (in SAS/GRAPH). The procedure offers both bivariate and spline interpolation methods for computing Z values.

group index
a numeric variable with positive integer values that correspond to values of a group variable. The index values are used to associate GraphData1 GraphDataN style elements with group values.

group variable
a variable in the input data set that is used to categorize chart variable values into groups. A group variable enables the data for each distinct group value to be rendered in a visually different manner. For example, a grouped scatter plot displays a distinct marker and color for each group value.

GRSEG
a SAS catalog entry that contains graphic output in a generic, rather than device-specific, format.

GSF
See graphics stream file.

handshaking
the exchange of signals between two devices over an interface for control or synchronization purposes. Data flow control is needed to ensure that data are not sent faster than the receiving device can process them. Handshaking usually involves sending signals between the device and the host computer in order to start and stop transmission of data.

hardcopy device
See graphics output device.

hardware character set
a set of character definitions held internally in a graphics output device. When a hardware character set is used, SAS/GRAPH software does not have to send the device all the commands to draw characters, only the corresponding character codes. Some devices have more than one hardware character set. *See also* font.

hardware handshaking
a method of data flow control in which the flow of data between the computer and device is regulated by signals sent over separate wires in the connecting cable. *See also* handshaking.

hatch
a fill pattern consisting of parallel lines at any specified angle.

HLS color model
a color-coding scheme that specifies a color in terms of its hue, lightness, and saturation components. Hue is the color, lightness is the percentage of white, and saturation is the attribute of a color that determines its relative strength and its departure from gray. Lightness and saturation added to the hue produce a specific shade.

host computer
a workstation or minicomputer accessed by a terminal or another workstation.

host font-rendering
a method of rendering fonts that relies on the capabilities of the operating environment.

HSV model
a color-coding scheme that specifies a color in terms of its hue, saturation, and value components. Hue is the color. Saturation is the aspect of a color that determines its relative strength and departure from gray. And value, or brightness, is the color's departure from black.

identification variable
a variable common to both the map data set and the response data set that the GMAP procedure uses to associate each pair of map coordinates and each response value with a unique map area.

image file
a file that contains bitmapped graphic information. Examples include GIF, PNG, TIFF, and JPEG files. Image files are a subset of graphics output files.

image format
a file format that displays a graphical representation. PNG, GIF, TIFF, and JPEG are examples of image formats, each with different characteristics.

image map
a diagram that associates graphic elements with HTML links to implement drill-down functionality. The graphic elements are represented by sets of coordinates. *See also* data tip.

import
to restore a SAS transport file to its original form (a SAS library, a SAS catalog, or a SAS data set) in the format that is appropriate for the host operating system. You use the CIMPORT procedure to import a SAS transport file that was created by the CPORT procedure.

inset
a graphical element such as a legend, line of text, or a table of text that is embedded inside of a graph's plot area.

interactive graph
output that features user controls such as menus, buttons, and pictures that a user can manipulate. The controls are driven by a Java applet or an ActiveX control.

interpolate
to estimate values that are between two or more known values.

interval axis
an axis where the distance between tick marks represents monotonically increasing or decreasing numeric units of some scale (like a ruler). The standard interval axis is called a LINEAR axis. Specialized interval axes include a TIME axis and a LOG axis.

Joint Photographic Experts Group
See JPEG.

JPEG (Joint Photographic Experts Group)
the name of an industry-standard file format for compressed images. Saving an image in JPEG format typically provides 10:1 compression with little perceptible loss in image quality.

justify
to position text in relation to the left or right margin or the center of the line.

key map
a SAS catalog entry used to translate the codes generated by the keys on a keyboard into their corresponding SAS/GRAPH internal character encoding. *See also* device map.

latitude
used with maps, the angular measure between the equator and the circle of parallel on which a point lies.

legend title
text that explains how to interpret the legend.

library reference
See libref.

libref (library reference)
a SAS name that is associated with the location of a SAS library. For example, in the name MYLIB.MYFILE, MYLIB is the libref, and MYFILE is a file in the SAS library. *See also* SAS library.

line printer daemon
under UNIX, the process that directs output to the system destinations indicated by the lp or lpr commands.

line property
a value that defines the pattern, thickness, or color of a line. By default, the value for a line property is derived from a style element in the current style. *See also* attribute bundle.

linear axis
an interval axis with ticks placed on a linear scale.

local selection mode
a feature that limits a selection to the graph or data table in which the selection is made, rather than applying the selection to all graphs and data tables associated with the project. *See also* global selection mode.

log axis
an axis displaying a logarithmic scale. A log axis is useful when data values span orders of magnitude.

longitude
used with maps, the angular measure between the reference meridian and the plane intersecting both poles and a point. The reference meridian, called the prime meridian, is assigned a longitude of 0, and other longitude values are measured from there in appropriate angular units (degrees or radians, for example).

macro variable reference
a string that contains the name of a macro variable that is referenced in order to substitute a value that is located or defined elsewhere.

major axis
in the graphics editor, the longest axis of a graphics object.

major tick mark
one of a series of points on an axis that mark the major divisions of the axis scale.

map
a graphic representation of an area. The area is often a geographic area, but it can also be any other area of any size. *See also* device map, key map.

map data set
a data set provided by SAS that contains variables whose values are coordinates that define the boundaries of map areas, such as a state or country.

mapping
the process of displaying data values on a map.

marker
a symbol such as a diamond, a circle, or a triangle that is used to indicate the location of, or annotate, a data point in a plot or graph.

marker property
a value that defines the symbol used as a marker, or its size, weight, or color. By default, the value for a marker property is derived from a style element in the current style. *See also* attribute bundle.

meridian
an imaginary circle of constant longitude around the surface of the earth perpendicular to the equator. *See also* parallel.

message area
the area in a window that displays messages from SAS.

metafile
a file, produced by the Metagraphics facility internal driver, that contains device-independent graphics commands in a special format. A user-written external driver routine is required to read and process the metafile.

Metagraphics driver
a type of SAS/GRAPH device driver that can be written by users. A Metagraphics driver consists of an internal driver (supplied with SAS/GRAPH software), which writes a metafile in a special format, and an external driver (written by the user), which decodes the metafile and writes device-specific commands.

metropolitan statistical area (MSA)
a geographic area defined by the U.S. Office of Management and Budget (OMB) to ensure consistency in the collection and tabulation of federal statistics. Each metropolitan statistical area includes at least one urban area with a minimum population of 50,000. Outlying counties with clear social and economic ties to the main urban area are also included.

midpoint
a value that represents the middle of a range of data values.

minor axis
in the graphics editor, the shortest axis of a graphics object.

minor tick mark
one of a series of points that fall between major tick marks on an axis scale.

MSA
See metropolitan statistical area.

name data set
a SAS data set in the map data library the GMAP procedure uses that contains both the names and identification numbers of the map areas, but not the coordinates that produce a map. *See also* map data set.

needle plot
a plot in which data points are connected by a vertical line that connects to a horizontal baseline. The baseline intersects the 0 value, or the minimum value on the vertical axis.

node
a connection point between two or more links. In a node/link diagram, nodes are typically represented as a box and enable you to access information and possibly to traverse the graph by drilling up or down in the structure.

numeric variable
a variable that contains only numeric values and related symbols, such as decimal points, plus signs, and minus signs.

ODS Graphics Editor
an interactive application that can be used to edit and annotate ODS Graphics output.

ODS template (graphics template)
a description of how output should appear when it is formatted. ODS templates are stored as compiled entries in a template store (item store). Common template types include STATGRAPH, STYLE, CROSSTABS, TAGSET, and TABLE.

offset
the distance between a graphics object's original position and its new position when it is moved. Offsets can be specified for legends, axes, an entire graph, or other graphics object.

opaque
a property of a background. Opaque backgrounds are filled with a color. Non-opaque backgrounds are transparent.

origin
in a three-dimensional graph, the point at which the X, Y, and Z axes intersect. In a two-dimensional graph, the point at which the X and Y axes intersect.

outlier
a data point that differs from the general trend of the data by more than is expected by chance alone. An outlier might be an erroneous data point or one that is not from the same sampling model as the rest of the data.

parallel
an imaginary circle of constant latitude around the surface of the earth parallel to the equator. *See also* meridian.

parameterized plot
a non-computed plot that requires parameterized data. The Graph Template Language offers several plots in both computed and parameterized versions, for example, BARCHART and BARCHARTPARM. Some computed plots such as REGRESSIONPLOT can be emulated with a SERIESPLOT if the input data represented points on a fit line.

parent block
when two or more blocks are nested, any layout block that contains one or more layout blocks is a parent of the contained blocks.

pattern type
in SAS/GRAPH software, the set of fill patterns that are valid for a particular type of graph. The PATTERN statement supports three pattern types: bar and block patterns, map and plot patterns, and pie and star patterns. *See also* fill pattern.

pie chart
a circular chart that is divided into slices by radial lines. Each slice represents the relative contribution of each part to the whole.

pipe
See unnamed pipe.

pixel
an element of an electronic image. A pixel is the smallest element on a display that can be assigned a separate color.

plot
a visual representation of data such as a scatter plot, needle plot, or contour plot.

plot line
the line joining the data points in a plot.

plotter
a class of graphics devices that typically use pens to draw hard-copy output.

PNG
See Portable Network Graphic.

polygon font
a SAS/GRAPH font in which the characters are drawn with enclosed areas that can be either filled or empty. *See also* stroked font.

polyline
in SAS/GRAPH software, a graphics object composed of connected line segments that might have attributes. A polyline is not a closed object; therefore, it cannot be filled with a pattern.

Portable Network Graphic (PNG)
a file format that returns the graphical output in separate files and that produces a non-interactive image. This format is similar to the GIF format, but has additional features, such as support for true-color images and better compression.

PostScript
a device-independent page description language for printing high-resolution integrated text and graphics.

predefined color
one of the set of colors for which SAS/GRAPH software defines and recognizes names (for example, BLACK, BLUE, and CYAN).

primary axis
the X or Y axis contrasted to the X2 or Y2 secondary axis.

primary plot
the plot in an overlay that determines axis features, such as axis type and axis label.

prism map
a three-dimensional map that uses prisms (polyhedrons with two parallel surfaces) of varying height to indicate the ordinal magnitude of a response variable.

procedure output area
the portion of the graphics output area where the output from a graphics procedure is displayed. *See also* graphics output area, data area.

projection
a two-dimensional map representation of unit areas on the surface of a sphere (for example, geographic regions on the surface of the Earth).

prompt character
a character sent by the host computer to a device to signal that the host has finished transmitting data and is ready for a response from the device.

protocol
a set of rules that govern data communications between computers, between computers and peripheral devices, and between software applications. TCP/IP, FTP, and HTTP are examples of protocols.

protocol converter
a device used to convert a standard or proprietary protocol of one device to the protocol suitable for the other device or tools in order to achieve interoperability. For example, a protocol converter enables ASCII terminals to communicate with host systems by converting the ASCII data stream to an IBM 3270 data stream and vice versa.

prototype layout
an overlay plot composite that appears in each cell of a classification panel. Each instance of the prototype represents a different subset (classification level) of the data.

radar chart
a chart that shows the relative frequency of data measures with statistics displayed along spokes that radiate from the center of the chart. The charts are often stacked on top of one another with circular reference lines, thus giving them the appearance of a radar screen. Variations of the radar chart have names based on what they look like; these include star charts, spider charts, wind rose charts, and calendar charts.

rasterizer
a device that accepts commands (such as moves and draws) as input and that converts those commands into a bit-map. Rasterizers are connected between host computers and graphics output devices that require bitmapped input.

regression analysis
an analysis that models a dependent (or response) variable as a function of one or more independent (or predictor) variables. The regression line, which is the set of predictions from the model, appears as a line or curve in a plot of the dependent variable against an independent variable.

relative coordinate
a coordinate that is measured from a point other than the origin. In the Annotate facility, this point is usually the endpoint of the last graphics element that was drawn. *See also* absolute coordinate.

replay
in SAS/GRAPH software, to display graphics output that is stored in a catalog entry using the GREPLAY procedure.

response data set
a SAS data set used by the GMAP procedure that contains data values associated with map areas and one or more identification variables. *See also* identification variable, response variable.

response levels
the individual values or ranges of values into which the GMAP or GCHART procedure divides the response variable. *See also* midpoint.

response value
any value of a response variable that the GMAP procedure represents on a map as different pattern/color combinations, or as raised map areas (prisms), spikes, or blocks of different heights. The GCHART procedure represents response values as bars, slices, spines, or blocks. *See also* midpoint.

response variable
in the GMAP procedure, a variable in the response data set that contains data values that are associated with a map area. *See also* chart variable, response data set, response levels.

RGB color model
a color-coding scheme that specifies a color in terms of amounts of red, green, and blue components.

RGBA color space
a color-coding scheme that specifies a color in terms of amounts of red, green, and blue components, along with an alpha channel that controls the color opacity. *See also* RGB color model.

role
a description of the purpose that a variable serves in a plot. For example, a series plot has predefined roles named for X , Y, GROUP, and CURVELABEL.

row axis
an external axis appearing on the left or right of a row of cells in a multi-cell layout.

row header
typically, the text that identifies the row contents in a multi-cell layout. This text can be aligned to the right or left of the cells in a row. The row header is not restricted to text (it can contain a plot or a legend, for example).

row major order
an order for populating cells of a layout or entries of a legend when the number of columns is specified. For example, in the default case: Start at the top left and fill cells or entries left-to-right. When the right-most column is filled, begin a new row below the previous row. Continue this until all content items have been placed in cells or entries. There might be empty cells/entries in the last row.

row weight
in a LATTICE layout, the proportion of height allotted to a specific row of the layout. The sum of all row weights is 1.

SAS catalog (catalog)
a SAS file that stores many different types of information in smaller units called catalog entries. A single SAS catalog can contain different types of catalog entries. *See also* SAS catalog entry.

SAS catalog entry (catalog entry)
an individual storage unit within a SAS catalog. Each entry has an entry type that identifies its purpose to SAS. *See also* entry type.

SAS data set (data set)
a file whose contents are in one of the native SAS file formats. There are two types of SAS data sets: SAS data files and SAS data views. *See also* descriptor information.

SAS date value (date value)
an integer that represents a date in SAS software. The integer represents the number of days between January 1, 1960, and another specified date. For example, the SAS date value 366 represents the calendar date January 1, 1961.

SAS function (function)
a type of SAS language element that is used to process one or more arguments and then to return a result that can be used in either an assignment statement or an expression.

SAS library
one or more files that are defined, recognized, and accessible by SAS, and that are referenced and stored as a unit. Each file is a member of the library.

SAS/GRAPH font
a font stored in the SASHELP.FONTS catalog, and a font created by the user and stored in a GFONTn catalog. These fonts can be used only by SAS/GRAPH procedures or other procedures that generate GRSEG output files. Examples of SAS/GRAPH fonts include Swiss, Simulate, and Marker. These fonts are provided for specialized purposes only.

scatter plot
a two- or three-dimensional plot that shows the joint variation of two (or three) variables from a group of table rows. The coordinates of each point in the plot correspond to the data values for a single table row (observation).

secondary axis
an X2 or Y2 axis, as contrasted with the primary axes X or Y.

segmented bar chart
a chart that consists of a grid and some vertical columns (bars). Each column is divided in order to represent subgroups of quantitative data.

sidebar
an area of certain multi-cell layouts external to the grid of cells where text or other graphical elements can appear. The LATTICE, DATAPANEL, and DATALATTICE layout support four sidebar areas (TOP, BOTTOM, LEFT, and RIGHT).

slider chart
a KPI chart characterized by a vertical or horizontal bar divided into qualitative ranges according to the boundary values that you specify. The actual value of the KPI is indicated with a triangle pointer on the chart.

snap
in the graphics editor, to automatically place graphics objects in the grid display area with precision.

sparse data
in classification panels with two or more classifiers, some crossings of the classification values might not be present in the input data. Such input data is called sparse data. By default, a DATAPANEL layout does not generate cells for sparse data, but if requested, it can produce empty cells as place holders for the non-existent crossings.

speedometer chart
a KPI chart characterized by a speedometer image with the tick marks evenly spaced around the dial, and colored ranges that correspond to the specified range boundaries. The actual value of the KPI is indicated by a long pointer. The target value, if it is specified, is displayed as a small, black triangle.

spine
a line on a star chart used to represent the relative value of the chart statistic for a midpoint. Spines are drawn outward from the center of the chart.

spoke
any of a number of lines that radiate from the center of a radar chart or a star chart. These lines represent statistical information.

standard deviation
a statistical measure of the variability of a group of data values. This measure, which is the most widely used measure of the dispersion of a frequency distribution, is equal to the square root of the variance.

star chart
a type of radar chart that features lines drawn to connect the chart statistics for each spoke, resulting in a star-like appearance.

stroked font
in SAS/GRAPH software, a font in which the characters are drawn with discrete line segments or circular arcs. *See also* polygon font.

style attribute
a visual property, such as color, font properties, and line characteristics, that is defined in ODS with a reserved name and value. Style attributes are collectively referenced by a style element within a style template.

style reference
a part of the Graph Template Language syntax that indicates the current value of a specific attribute of a specific style element. For example, SIZE=GraphTitleText:FontSize means to assign to SIZE the value of the FontSize attribute of the GraphTitleText style element from the current style.

subgroup variable
the variable in the input data set for a chart that is used to proportionally fill areas of the bars or blocks on a bar chart, or to identify separate rings of a pie chart.

summary variable
a variable in an input data set whose values some SAS/GRAPH procedures total or average to produce the sum or mean statistics, respectively.

surface map
a three-dimensional map that uses spikes of varying heights to indicate levels of relative magnitude.

surface plot
a three-dimensional graph that displays values of a vertical Z variable based on gridded X and Y variables.

system font
a font that can be used by any SAS procedure and by other software such as Microsoft Word. These fonts include TrueType and Type1 fonts. Examples of system fonts include Albany AMT, Monotype Sorts, and Arial.

Tagged Image File Format
See TIFF.

template
a specification of an area or areas on a page. A GREPLAY template defines a layout in which you can display one or more graphs on a single page.

template compile time
the phase when the source program of a template definition is submitted. The syntax of the definition is evaluated for correctness. If no errors are detected, the definition is converted to a binary format and stored for later access.

template panel
in the GREPLAY procedure, a part of the template in which one or more graphics can be displayed. A template can contain one or more panels.

template run time
the actions performed when a compiled template is bound to a data object and then rendered to produce a graph. Run-time errors can occur that prevent a graph from being produced.

text property
any of a common set of characteristics that can be specified for any text string: color, family, size, weight, and style. By default, values for these properties are derived from a style element in the current style. *See also* attribute bundle, style attribute.

thumbnail
a small image that can be selected in order to display a larger image.

TIFF (Tagged Image File Format)
An industry-standard file format for storing compressed images. The Tagged Image File Format specifies compression routines and file formats for a variety of image types, including bilevel, grayscale, and color.

TIGER
a format for map data that was developed by the United States Census Bureau. As of 2007, the TIGER Record Type (RT) file format has been superseded by the TIGER shapefile format. The acronym means Topologically Integrated Geographic Encoding and Referencing.

tile chart
See treemap.

tilt angle
the measure in degrees from the horizontal axis to the major axis of an object.

time axis
an axis type that displays only SAS date, time, or datetime values. Axis tick value increments can be specified as time or date intervals, such as MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, or YEAR.

title area
the region above the graph area where text produced by ENTRYTITLE statements appears.

traffic light chart
a KPI chart in the form of a traffic light that contains one light for each qualitative range. The range that contains the actual value is displayed in a specified color. The remaining ranges are gray. In other words, only one “light” is “turned on” at a time. Traffic lights do not display target values.

transformation
in the DATA Step Graphics Interface (DSGI), a mapping of the window coordinates to the viewport coordinates.

translate
to change the location of a graphics object.

treemap (tile chart)
a graph that represents the relative values of data by using nested rectangular areas. The color or pattern of each area represents the value of one measure in the query. The size of each area represents the value of the another measure in the query.

typeface
a set of characters with a common design, represented by one or more fonts that differ in weight, orientation, width, size, and spacing. For example, Arial, Arial Bold,

Arial Italic, and Arial Bold Italic share the same typeface (Arial), but differ in their orientation and weight.

unit
a single quantity of measurement. In SAS/GRAPH software, units can represent any of the following: centimeters, percentages, points, inches, or cells.

unnamed pipe (pipe)
under UNIX operating systems and derivatives, the facility that links one command to another so that the standard output of one becomes the standard input of the other.

variable label
descriptive text that is associated with a variable, and that can be printed in the output by certain procedures. By default, this text is the name of a variable or of a label previously assigned with the LABEL= option.

view
a definition of a virtual data set that is named and stored for later use. A view contains no data; it merely describes or defines data that is stored elsewhere.

wall
the area bounded by orthogonal axis pairs. In two-dimensional graphs, there is one wall bounded by the XY axes. In three-dimensional graphs, there are three walls, bounded by the XY, YZ, and XZ axes. A wall has an optional outline and can be opaque or transparent.

web server
a computer program that delivers (serves) content, such as web pages, over the World Wide Web. It can also refer to the computer or virtual machine that runs the program.

weight variable
a numeric variable that represents a weight (for example, costs) to be applied to observations.

wind rose chart
a type of radar chart that depicts how wind speed and direction are typically distributed at a particular location. The cardinal directions or subdirections with the longest spokes indicate the wind direction that has the greatest frequency. *See also* radar chart.

XON/XOFF handshaking
a method of data flow control in which the flow of data between a computer and a device is regulated by the transmission of XON (DC1) and XOFF (DC3) control characters between the device and the computer.

Index

B

C

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z