

Wow! You Did That Map with SAS/GRAPH®?

Louise S. Hadden, Abt Associates Inc., Cambridge, MA

Mike Zdeb, University@Albany School of Public Health, Rensselaer, NY

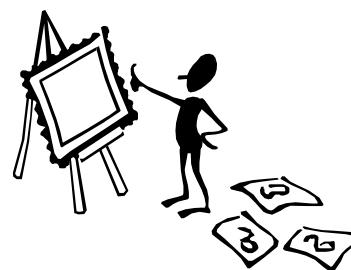
Robert Allison, SAS Institute Inc., Cary, NC

ABSTRACT

SAS/GRAPH and ODS provide SAS programmers with tools to create customized, full color maps for a variety of uses. The production of useful, impressive looking maps using SAS/GRAPH is surprisingly easy, and can be accomplished with greater ease in SAS 9.2. This paper walks through the creation of a number of different maps using SAS 9.1 and 9.2 and shows some exciting new choices and possibilities in SAS 9.2. Examples include: PROC MAPIMPORT (from shapefile to map data set); tidbits from SAS MAPSONLINE including SASHELP.ZIPCODE, SASHELP.MAPFMTS and WORLDCTS; new maps available in 9.2 (Cuba, Tasmania and others); %colorscale and Colorbrewer (choosing map colors for data-driven mapping in 9.1, ODS styles in 9.2); improved "look and feel" in 9.2 (increase in default color list, true type fonts, better image clarity); distance calculations available in 9.1 and 9.2 (a macro in 9.1, GEODIST and ZIPCITYDISTANCE functions in 9.2); and last but certainly not least examples of PROC GMAP+GCHART (bar chart overlays on a map with PDF in 9.1, transparent GIF in 9.2).

INTRODUCTION

The purpose of this paper is to demonstrate how easy it is to produce spectacular, colorful and informative maps using SAS/GRAPH. Maps created with 9.1 and 9.2 will be presented side by side, with a discussion of the tools and techniques used to create each map including code "snippets" (full code will be available in a zip file in the NESUG Proceedings.) All sample code is provided "as is." SAS 9.2 is currently available, but if you do not have access to it yet, it is important to note that these maps CAN be created using 9.1 and 8.2. Many of the tools and map data sets that are being shipped with 9.2 are available for download for earlier versions.



PROC MAPIMPORT

PROC MAPIMPORT (available in SAS 9.x) allows SAS programmers to import ESRI shapefiles and create SAS map data sets for geographic areas that SAS does not provide. There are a number of sources for free shapefiles on the web. A Google search for "free shapefiles" is a good place to start looking. If you would rather not search, a few good resources are...

ESRI (a GIS software company)

http://arcdata.esri.com/data/tiger2000/tiger_download.cfm

US Census Bureau

<http://www.census.gov/geo/www/cob/index.html>

CDC (Centers for Disease Control)

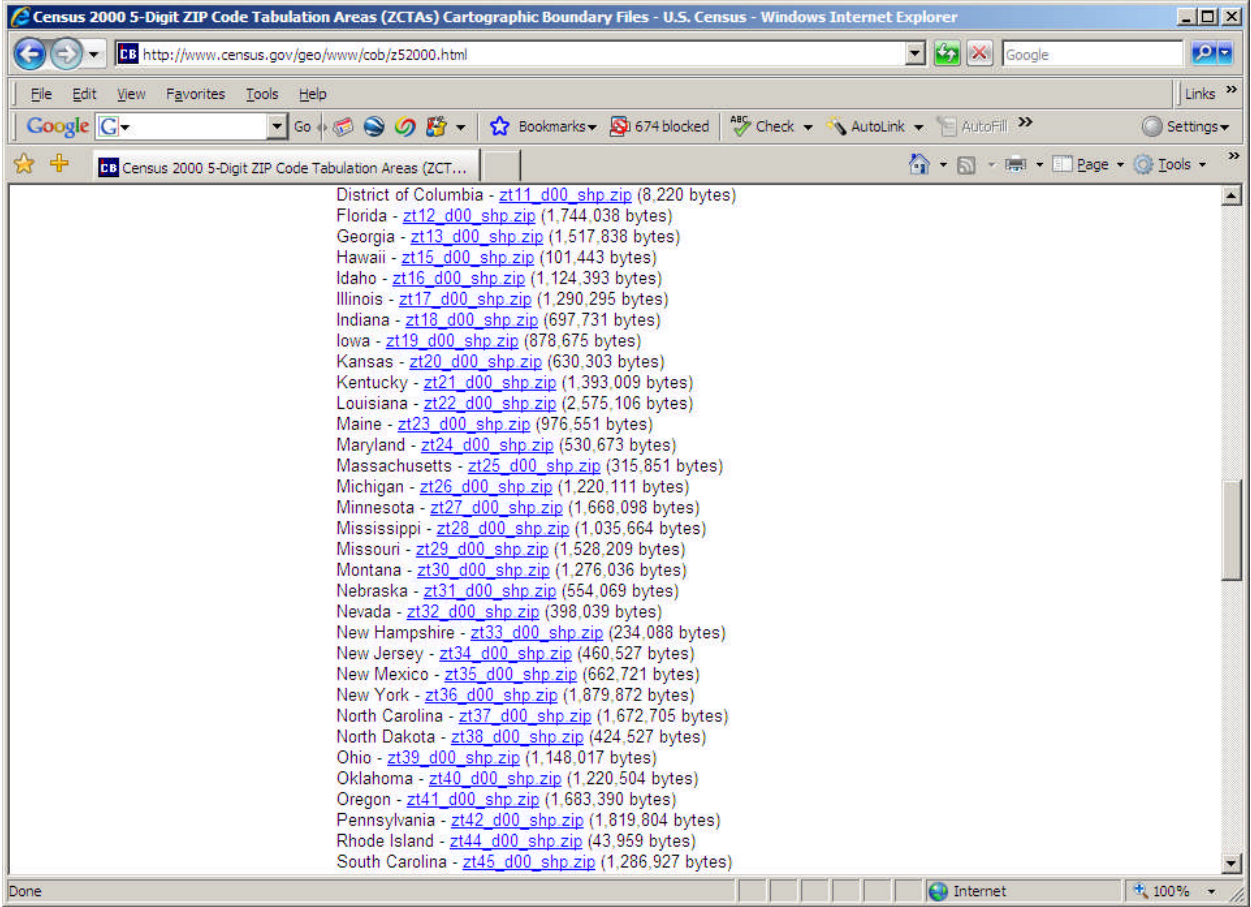
<http://www.cdc.gov/epiinfo/shape.htm>

CIP (International Potato Center, part of the Consultative Group on International Agricultural Research)

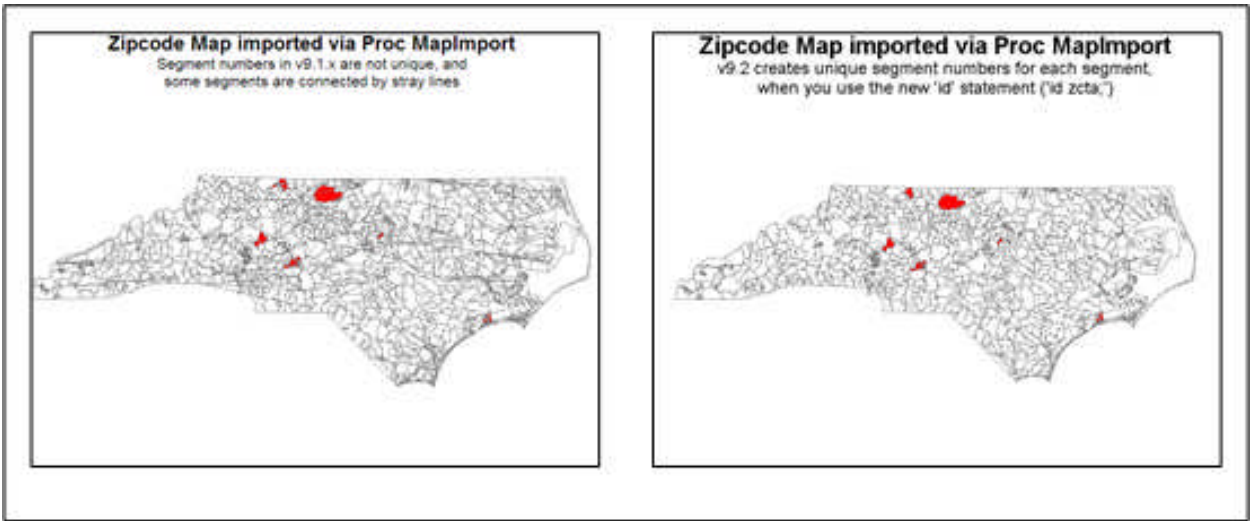
<http://research.cip.cgiar.org/gis/index.php>

A common use of PROC MAPIMPORT is for US Postal Zip Code (or really ZCTA) maps. ZCTA maps are available from the Bureau of the Census, and in conjunction with the SASHELP.ZIPCODE file described below and other data sources, allow the graphical display of US Postal Zip Code level data. Our first examples are of using PROC MAPIMPORT to create a US Postal Zip Code map for the state of North Carolina.

A FREE Zip Code (or more accurately ZCTA) ArcView Shapefile for the state of North Carolina was downloaded from the Bureau of the Census Cartographic Boundary Files web page (see US Census Bureau link above.)



SAS's PROC MAPIMPORT was used to import the zip code boundary maps for the entire state. The resulting maps for 9.1 and 9.2 are shown below.



```
/* Version 9.1 */
proc mapimport
  datafile="./zt37_d00.shp"
  out=mymap91;
run;
```

```
/* Version 9.2 */
proc mapimport
  datafile="./zt37_d00.shp"
  out=mymap92;
  id zcta;
run;
```

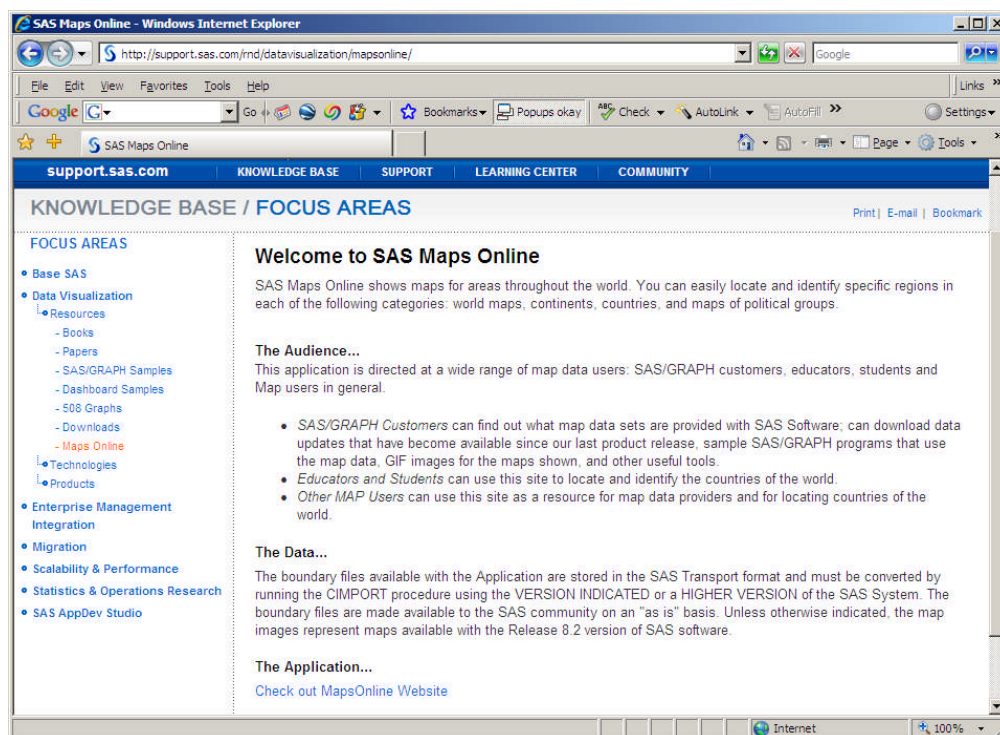
In the SAS-provided map data sets, if there are areas (polygons, such as states, counties, etc.) that have more than one piece, each piece has a different 'segment' number in the map data set, and PROC GMAP knows to "lift the pen" when drawing between each segment (i.e., the separate segments are not connected with a link.) For example, the multiple islands of the state of Hawaii in the MAPS.US data set are not connected with lines. In 9.1, PROC MAPIMPORT generally made all the map areas have the same 'segment' since it had no concept of which imported variable was the 'id' variable. Therefore, as demonstrated in the 9.1 example above, when there were multiple areas of the map with the same 'id', PROC GMAP draws a line between them, since it doesn't know they are separate and distinct segments of the same area. This results in the stray lines which make the 9.1 example look a little 'cluttered.' To work around this problem in 9.1, you can post-process the imported map data set, and change the segment numbers so that they are unique for each piece. However, it is hard to guarantee that the map will turn out right.

In 9.2, PROC MAPIMPORT was enhanced to allow the user to specify an 'id' variable (similar to specifying an id in PROC GPROJECT and PROC GMAP.) Now that PROC MAPIMPORT knows which variable is the id, it can intuit how you want to draw the map, and therefore assign segment numbers that are unique for each segment. Now the 9.2 Zip Code map doesn't have the stray lines that were present in the 9.1 version.

MAPSONLINE

MAPSONLINE is a SAS-provided web page with a number of useful tools regarding mapping, including map data sets and updates, related data sets (such as SASHELP.ZIPCODE and WORLDCTS), and sample programs.

<http://support.sas.com/rnd/datavisualization/mapsonline/>



The examples shown in this paper use the SASHELP.ZIPCODE, SASHELP.MAPFMTS, WORLD, and WORLDCTS

data sets. It is important to note that the updates in MAPSONLINE are NOT VERSION SPECIFIC. The files will work with versions 8.2 and up. They are provided in CPORT transport files to facilitate use across versions and platforms.

SAS 9.2 ships with the latest versions of the SASHELP.ZIPCODE file, SASHELP.MAPFMTS file, and all SAS-provided maps. These files do change over time (as shown in the first example below) so it is important to visit MAPSONLINE on a regular basis to download newer versions of files, or, you may wish to download an older file which matches the timing of your response data. For example, the SASHELP.ZIPCODE file is updated on a quarterly basis, with the latest version as of the writing of this paper posted in April 2008. The latest and greatest SASHELP.ZIPCODE and SASHELP.MAPFMTS files are obtained under 'Misc Updates.'

SAS Maps Online - Windows Internet Explorer

http://support.sas.com/rnd/datavisualization/maponline/html/misc.html

File Edit View Favorites Tools Help

Google

Go

Bookmarks

333 blocked

ABC Check

AutoLink

AutoFill

Send to

Settings

SAS Maps Online

support.sas.com > Communities > Data Visualization Community

Navigate our Site

Maps Online Downloads > Misc. Updates

Home Maps What's New Downloads Resources

Feedback FAQ

Version 6.12 Maps

Version 8.2 Maps

Country Map

Data Updates

Misc Updates

GIS US State Boundary Data

Geocoding

Note: After downloading and unzipping these tables you must take them out of transport format by running the [CIMPORT procedure](#) using the VERSION INDICATED or a HIGHER VERSION of the SAS System.

Name	Description	Date Added	SAS Version
Zipcode Dataset	New zipcode dataset, for July 2007, contains 41,763 observations and 18 variables. Includes a new dataset ZIPMIL containing APO and FPO zips. Source: zipcodedownload.com July2007.	Jul2007	V8
Zipcode Dataset	Discrepancies in lat/longs updated, for April 2007. Also includes a new dataset ZIPMIL containing APO and FPO zips. Source: zipcodedownload.com June2007.	Jun2007	V8
USCITY Dataset	Missing cityfips for Jacksonville, FL added Note: Although this dataset will be officially released with SAS Version 9.2, this copy is stored in a version 8 format that is accessible to SAS V8 users and beyond.	July2007	V8
US, USCENTER, USCITY Datasets	- The 3 datasets were re-projected when USCITY was updated - Variable STATECODE added - Data added for Puerto Rico Note: Although this dataset will be officially released with SAS Version 9.2, this copy is stored in a version 8 format that is accessible to SAS V8 users and beyond.	Apr2007	V8

Done

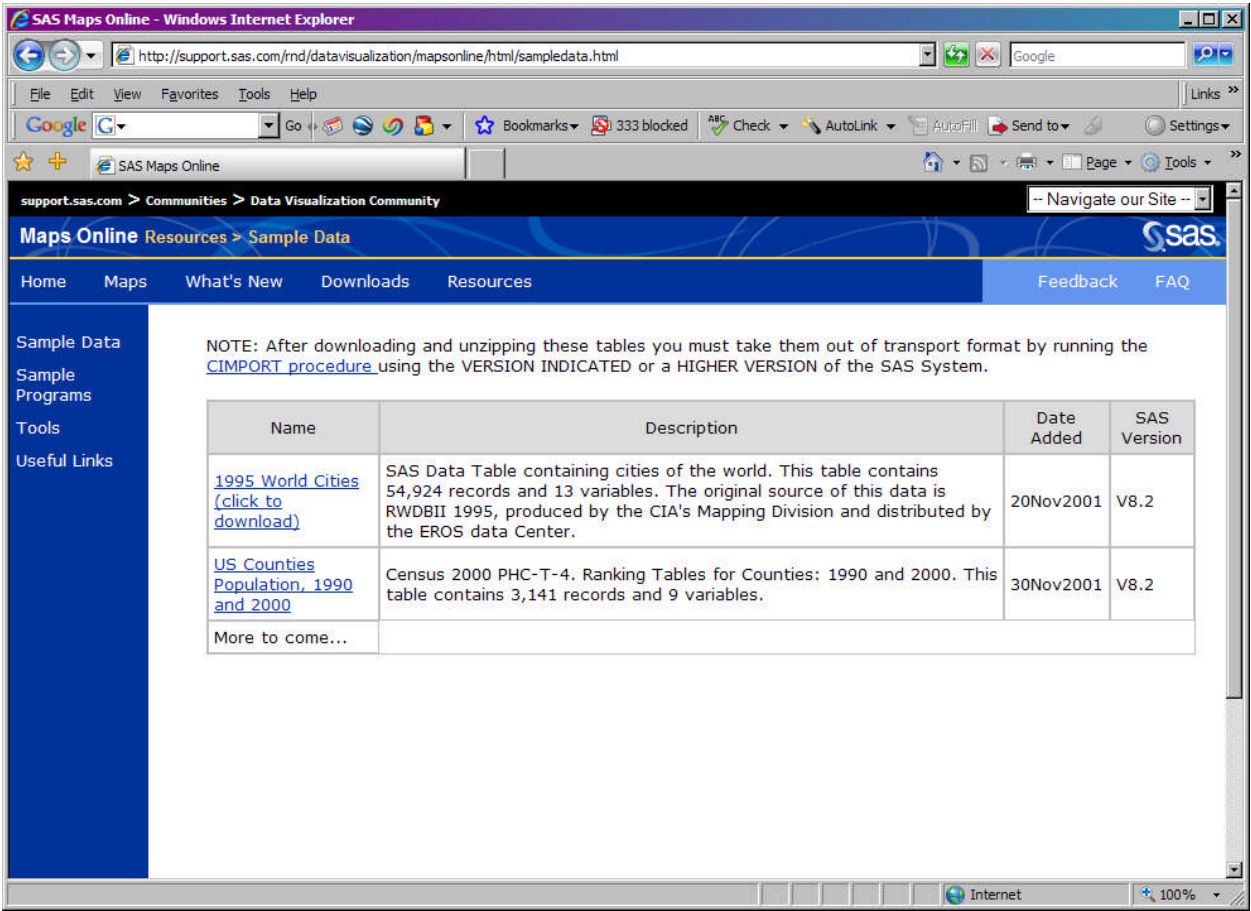
Internet

100%

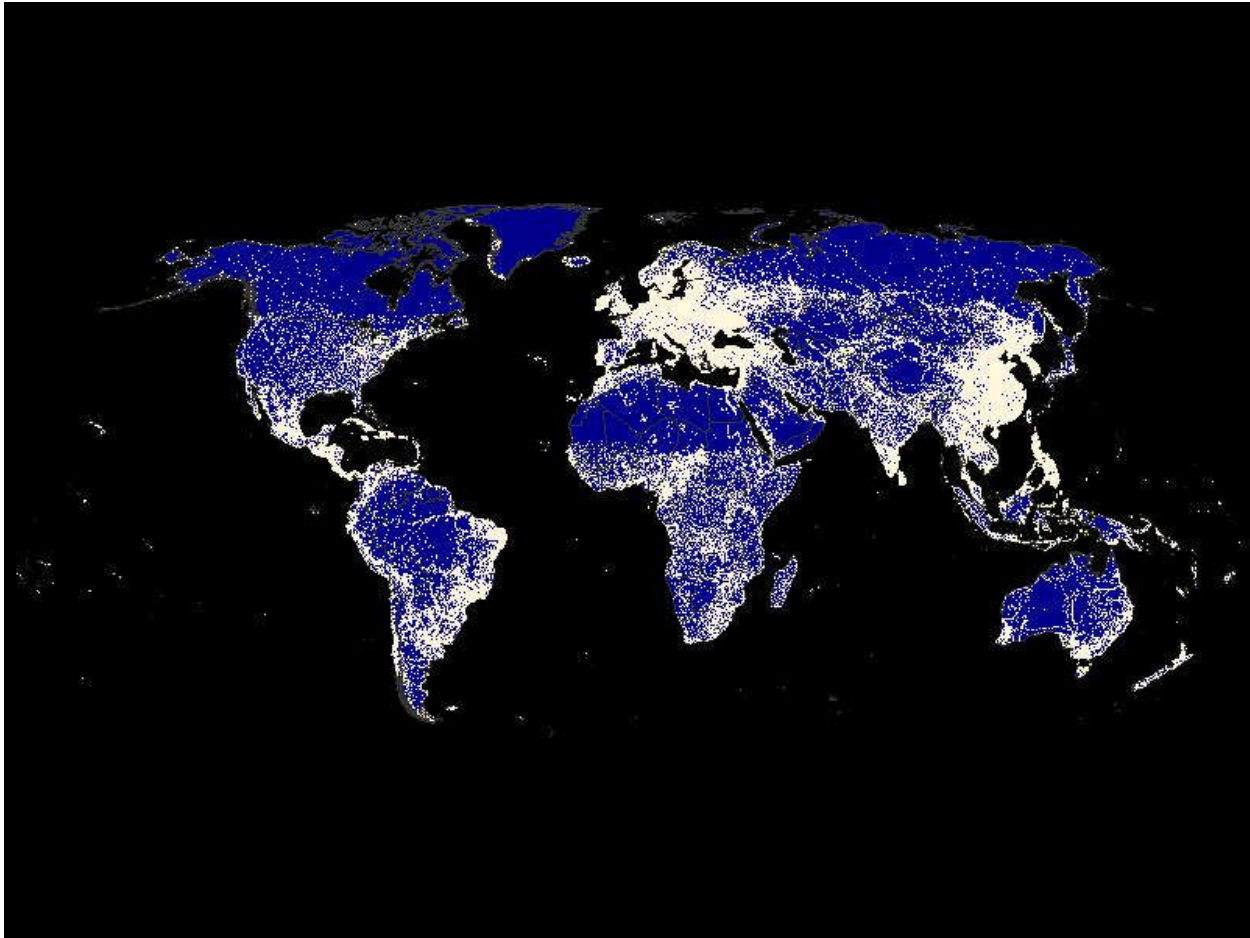


As can be seen in the examples above, Zip Codes are occasionally reassigned by the US Postal Service, just to reinforce the need for maintaining an updated version of the SASHELP.ZIPCODE file. The two Wake County maps demonstrate how Zip Code 27523 was used for the city of Eagle Rock (central Wake County, NC) a few years ago, and as of the writing of this paper, it is used for the city of Apex (in western Wake County, NC.)

Our second MAPSONLINE example uses the WORLD SAS map data set and the WORLDCTS (world cities) data set. The WORLD data set will be present in your current SAS maps directory, but should be updated with the version in MAPSONLINE. The WORLDCTS data set is available under Resources in MAPSONLINE.



The example below is created by creating an annotate data base with light dots representing world cities sized according to population overlaid on a world map (excluding Antarctica.) The world map is created from the SAS-supplied world map data set, while the latitude, longitude, and population of world cities comes from the SAS-supplied WORLDCTS (world cities) data set. There is no need to wait for version 9.2 to create this, or similar, data sets. The WORLDCTS data set is available for download from MAPSONLINE now. This technique offers an excellent way to represent various statistics on a world-wide basis.



NEW MAPS AVAILABLE FOR 9.2 (AND NOW ON MAPSONLINE!)

As noted above, SAS map data sets are updated on a regular basis. For 9.2 (and available now for free download from MAPSONLINE) an updated WORLD map data set is available. This data set includes Cuba and Tasmania, which were missing in the earlier version of the WORLD map data set. Please check MAPSONLINE regularly for updates to your favorite map data sets. Apart from the WORLD map data set, Germany, Switzerland, Serbia, Montenegro, and China are among the map data sets that have been updated in the past year.



CHOOSING MAP COLORS

SAS provides programmers with many options to use color to enhance SAS output. In addition, there are other valuable resources to aid color choices and specifications while using SAS procedures. Ordinarily, maps (and graphs) produced by SAS/GRAPH utilize colors and patterns in default lists unless specifically directed otherwise. SAS programmers can specify their own color list, and/or specify a list of patterns. Colors can be expressed in a number of different ways, including color name, RGB or Hex value, and HLS Value. Currently, there are a number of ways to choose a color scheme for SAS maps. Colors for each "pattern" or "level" to be represented can be chosen by hand; a tool such as Colorbrewer can be used to select colors; or %colorscale, a SAS-supplied macro, can be used. If you choose by hand or use %colorscale, a color chart such as the one shown below (from SAS TS-688) is useful.

Name	RGB Value	HLS Value	Sample	Description
BLACK	CX000000	H0000000		Black
Navy	CX000080	H00040FF		
DarkBlue	CX00008B	H00046FF		
MediumBlue	CX0000CD	H00067FF		
BLUE	CX0000FF	H00080FF		Blue
DarkGreen	CX006400	H0EF32FF		
Green	CX008000	H0EF40FF		
Teal	CX008080	H12C40FF		
DarkCyan	CX008B8B	H12C46FF		
DeepSkyBlue	CX00BFFF	H13980FF		
DarkTurquoise	CX00CED1	H12C59FF		
MediumSpringGreen	CX0FA9A	H1157DFF		
GREEN	CX00FF00	H0F080FF		Green
SpringGreen	CX00FF7F	H10D80FF		
CYAN	CX00FFFF	H12C80FF		Cyan
BRBL	CX010101	H0980101		Brownish black
VIB	CX007055	H00136DF		Vivid blue
VDEYG	CX0A260E	H0F81894		Very deep yellowish green
DEB	CX100F25	H0011B6D		Deep blue
VIG	CX118044	H10C48C3		Vivid green
VDAGB	CX121519	H14E162D		Very dark greenish blue
VDAG	CX121915	H10C162D		Very dark green
VDABG	CX121919	H12A162D		Very dark bluish green

In the code sample below (choosing colors by hand), the response variable to be mapped has four levels, so four pattern statements are supplied. Colors in this case are specified using names and abbreviations for names, but could have been specified using RGB values, HLS values and Hex values.

```
/* define patterns */
pattern1 value=msolid color=vpag; /* abbreviation for very pale green */
pattern2 value=msolid color=vpab; /* abbreviation for very pale blue */
pattern3 value=msolid color=pink;
pattern4 value=msolid color=yellow;
```

The COLORSCALE macro (available for download on the support.sas.com website) can be used to determine a list of colors in a gradient. The TOP and BOTTOM colors are required while a middle color is optional. The value N sets the number of intermediate colors. The macro takes the following parameters:

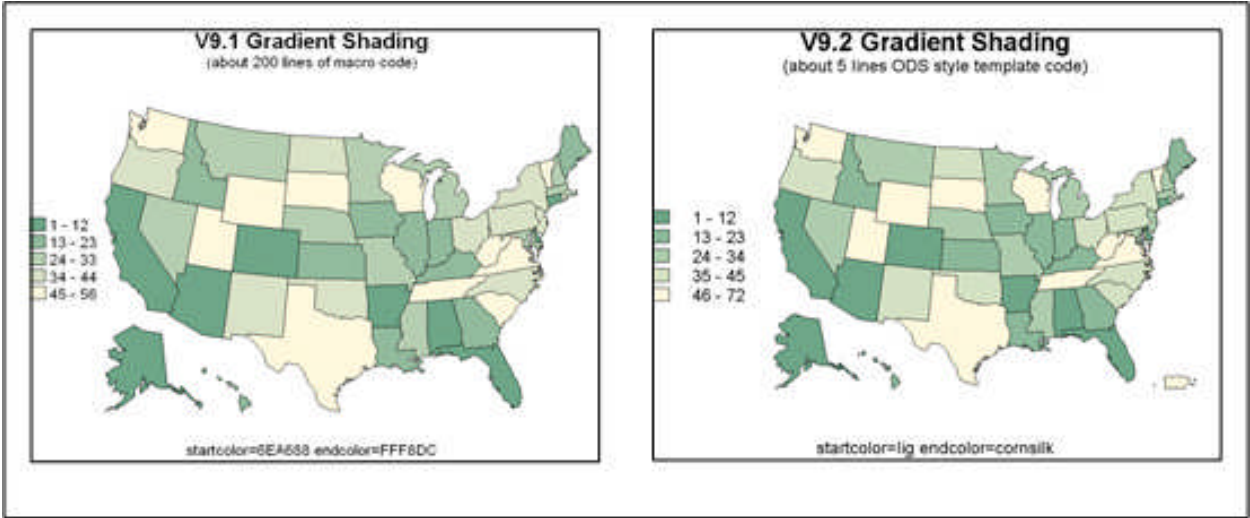
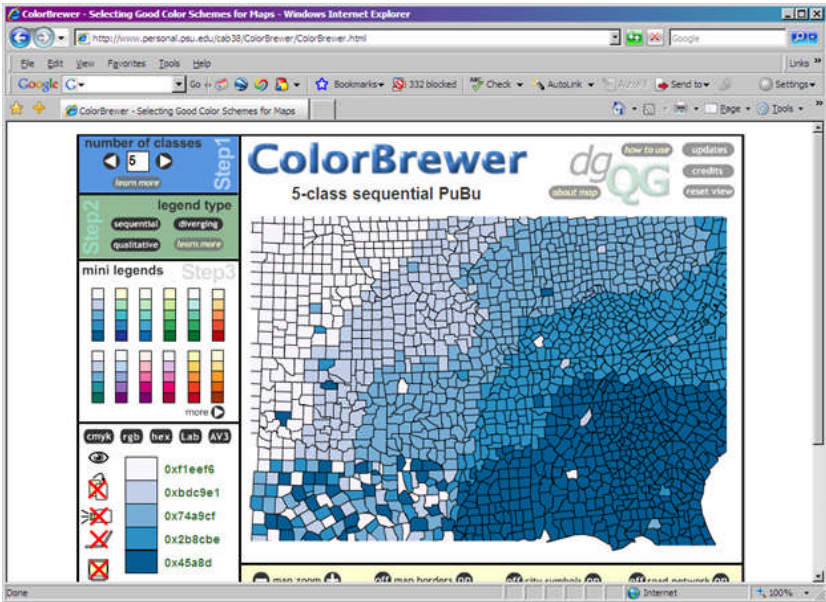
- TOP: color displayed on top of the output
- MIDDLE: optional middle color; the gradient forced through this color
- BOTTOM: color displayed on the bottom of the output
- N: the number of intermediate colors
- DSN: name of the dataset that stores the colors.
- SWATCH: if "Y", display a sample of the colors.

Colors should be represented as RGB hex values, such as FFFFFFFF for white or 000000 for black. The color values needed for this macro are the last 6 digits of the RGB values from a chart such as the one shown above. A sample macro call is shown below. This would create a dataset called 'anno' which stores 5 colors ranging from white to black. The %colorscale macro is around 200 lines of macro code, which must be available either through prior invocation of the macro or in a stored macro library.

```
%colorscale(000000,,FFFFFF,3,anno);
```

It is still handy to have a color chart around to choose the beginning, intermediate (if desired) and end colors.

Colorbrewer is a wonderful (free) website that allows you to choose color schemes “online.” One can choose the number of levels (in the screenshot shown below, 5.) Then choose legend type (in this case, sequential.) The “step 3” box then offers a number of options for color schemes (a particularly attractive blue gradient scheme was chosen in this case.) Directly below one can click on any number of color representation codes (in this case, HEX is shown.) These codes can then be used in pattern statements as shown above. Colorbrewer is particularly handy if you will be presenting maps online as you can see how the colors will look viewed online. There are many more features to Colorbrewer than can be described here: a visit to the website is well worth the time (<http://www.colorbrewer.org>.)

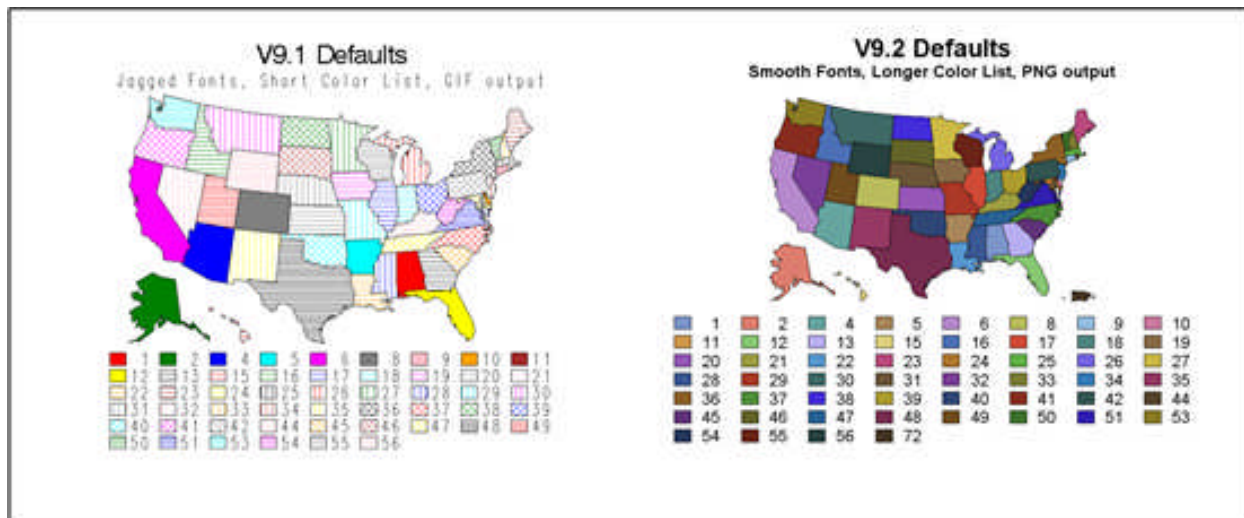


The 9.1 map above was created using %colorscale (but could also have been created choosing colors by hand or by using Colorbrewer.) In 9.2, ODS styles are honored by the ‘server-side’ graphics devices (such as device=png, gif, win, xcolor, etc.) and you can use the ODS styles to do gradient shading in PROC GMAP. (In fact, the default for PROC GMAPs using ‘levels=’ is to use gradient shading in 9.2.) All that is required to choose your start and end colors is to modify the ODS template, as below.

```
proc template;
define style styles.grade;
  parent=styles.listing;
  style twocolorramp / startcolor=lightgreen endcolor=cornsilk;
end;
run;
```

Note that color names (such as cornsilk) and abbreviations (such as lig) can be used as opposed to using HEX codes in %colorscale.

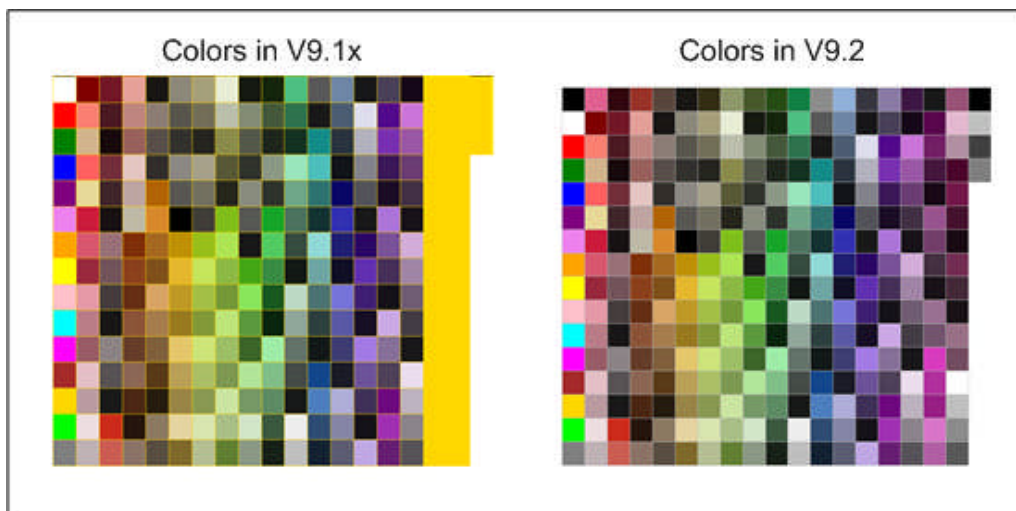
SAS/GRAPH DEFAULTS AND COLORS



In 9.1 and earlier, SAS/GRAPH used SAS/GRAPH software fonts by default. These fonts were the only ones guaranteed to be available on all systems (Windows, Unix, Mainframe, etc). These fonts do not support anti-aliasing (smoothing of the edges), and look somewhat blocky/jagged/stair step along curved edges of the letters. Also, the default color list had about a dozen colors in it, and once a map or chart ran out of those colors in its patterns it would cycle through the same colors, but use a different hatch-mark pattern.

In 9.2, nice smooth-edge fonts are used by default. These fonts are rendered with anti-aliasing, so they have smoothed edges (rather than jagged/grainy/stair step edges). Rather than having a short color list and cycling those colors through several cross-hatch patterns, a much longer color list is used in 9.2, and shades of those colors are cycled through. You can always override the default fonts and colors, by specifying the fonts and colors you want - but these new defaults help you get a much nicer-looking GMAP "out of the box".

Note that this particular example is a bit extreme - it would not normally be useful to plot each of the 50 states a different color. This is "for demonstration purposes only!" :)



Prior to 9.2, SAS/GRAPH could only support 256 colors in a single graph. 256 might seem like a large number, but remember that this not only includes the colors that you specify, but also the colors needed by SAS/GRAPH to do things like anti-aliasing to make smooth-edged text (which basically uses several shades of the text color to fill in the jagged edges around curved font surfaces.) Also, if an image or picture is used in the background this can consume a lot of colors. A lot of colors are also used for things like the gradient shading on the sides of bar charts with cylinder shaped bars.

The example above shows all the “named” SAS colors on one page. In 9.1, since only 256 colors can be shown and there are more than 256 named SAS colors, you run out of colors towards the end. When you run out of colors, the results are somewhat unpredictable. SAS simply reuses some of the already-mapped colors, rather than the specified named color, when it runs out. In 9.2, an unlimited number of colors (more than 16 million) can be used in a single graph. Note that you must use a device that supports more than 256 colors, such as device=png. The standard in 9.1, device=gif, does not support more than 256 colors. For this reason, the new default device for SAS/GRAPH is device=png in 9.2 rather than device=gif in 9.1.

There are several default changes and other enhancements that require more than 256 colors in 9.2:

- Nice smooth-edged/anti-aliased fonts are used by default, rather than the old jagged-edged SAS/GRAPH software fonts.
- Longer default color lists – rather than repeating the same colors with different patterns once a color list was exhausted, in 9.2 the colors in the color list are repeated in various shades.
- ODS graph styles are honored in all devices in v9.2 (previously they were only supported by device=java and activex). These graph styles frequently use images in the backgrounds, and some of these images consume a lot of colors.

DISTANCE CALCULATIONS

The SASHELP.ZIPCODE described above can be used to enable SAS-written functions, to develop user-defined formats, calculate distances between Zip Codes and to annotate SAS/GRAPH maps with information in the file. It is important to note if you do not have a current ZIPCODE file in your SASHELP subdirectory, these uses are not possible!

9.1 SAS-written functions utilizing the SASHELP.ZIPCODE file include ZIPCITY, ZIPSTATE, ZIPNAME, ZIPNAMEL and ZIPFIPS. For example, the ZIPCITY function takes ZIPCODE as its argument and returns a proper case city name and a two character postal state abbreviation.

```

ZIPCITY('02138') returns 'Cambridge, MA'
ZIPSTATE('02138') returns 'MA'
ZIPNAME('02138') returns 'MASSACHUSETTS'
ZIPNAMEL('02138') returns 'Massachusetts'
ZIPFIPS('02138') returns 25 (FIPS state code for Massachusetts)

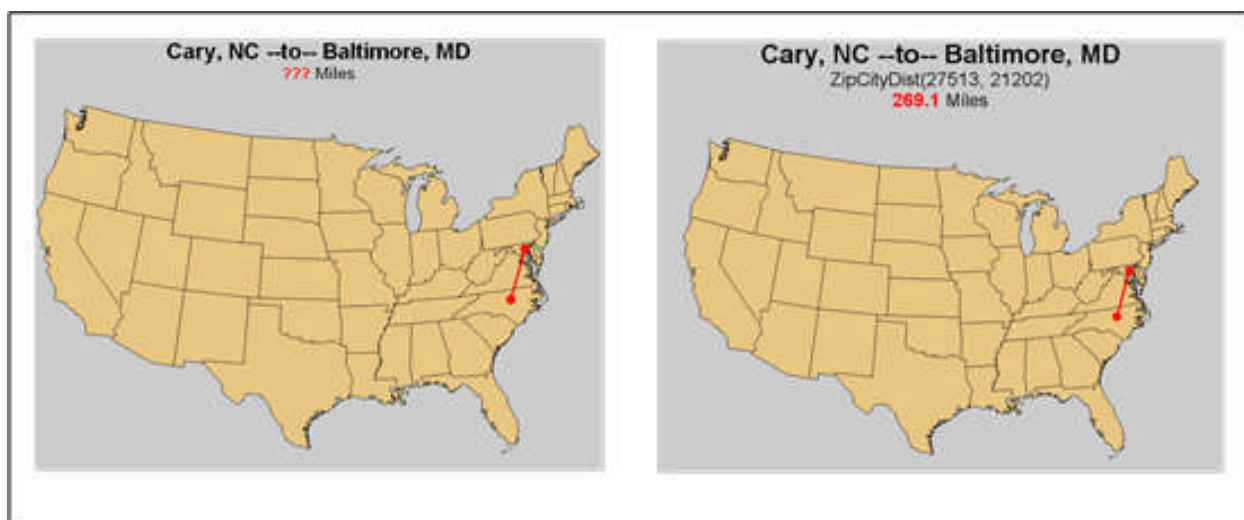
```

The SASHELP.ZIPCODE file can also be used to calculate distances between Zip Code centroids in 8.2 and 9.1. support.sas.com has provided an example of this use at <http://support.sas.com/techsup/unotes/SN/005/005325.html>.

```

/* calculate the distance between zip code centroids */
%macro geodist(lat1, long1, lat2, long2);
%let pi180=0.0174532925199433;
7921.6623*arsin(sqrt((sin((&pi180*&lat2-&pi180*&lat1)/2))**2+
cos(&pi180*&lat1)*cos(&pi180*&lat2)*(sin((&pi180*&long2-
&pi180*&long1)/2))**2));
%mend;

```



In 9.2, SAS builds on this functionality by adding new functions and procedures. In 9.1, SAS did not provide any functions to calculate distance between longitude and latitude pairs, but this changes with the addition of the ZIPCITYDIST() and GEODIST() functions in 9.2. ZIPCITYDIST() calculates the number of miles between 2 US Zip Codes. GEODIST() allows the user to specify 2 sets of longitude and latitude coordinates, and calculates the distance between the pairs. You can specify the units for the results, such as miles or kilometers. ZIPCITYDIST() only works on US Zip Codes, while GEODIST() can be used for coordinates anywhere around the globe. The example above shows the use of both new functions.

```

%let zip1=27513;
%let zip2=21202;

/* latitude and longitude coordinates */

proc sql;
create table zip1 as select * from sashelp.zipcode where zip=&zip1;
create table zip2 as select * from sashelp.zipcode where zip=&zip2;
select x into :long1 from zip1;
select y into :lat1 from zip1;
select x into :long2 from zip2;
select y into :lat2 from zip2;
quit; run;

%let city1=%sysfunc(zipcity(&zip1));

```

```
%let city2=%sysfunc(zipcity(&zip2));

/* two new 9.2 functions */

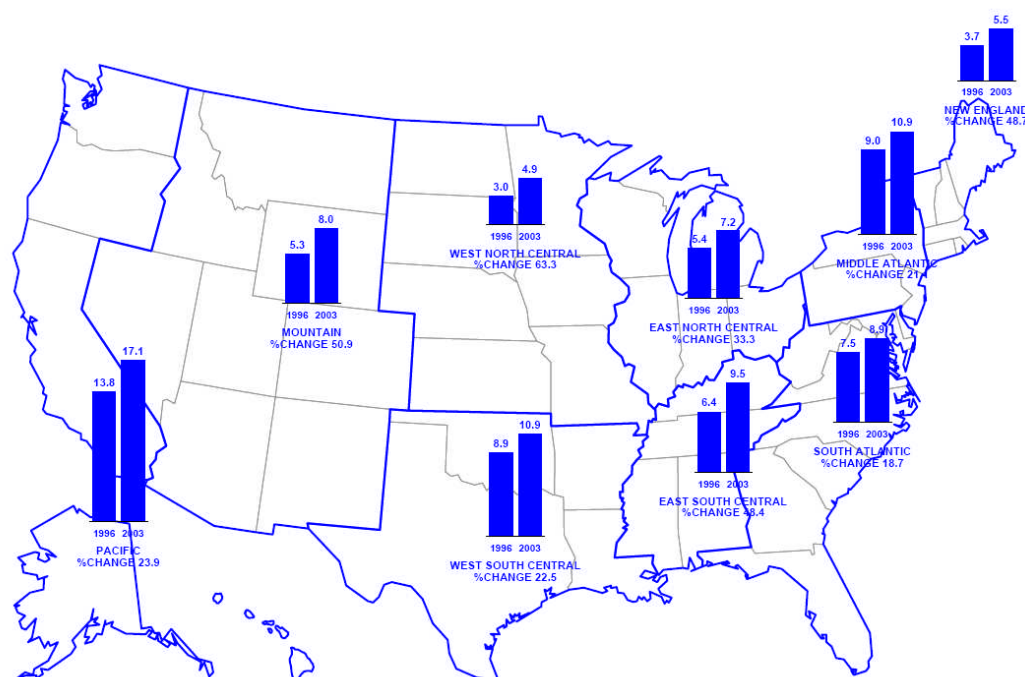
%let zipdist=%sysfunc(zipcitydistance(&zip1,&zip2));

%let geodist=%sysfunc(geodist(&lat1, &long1, &lat2, &long2, 'M'),comma10.5);
```

GRAPHIC OVERLAYS

There are various ways of using SAS/Graph to create output with the multiple areas of charts, maps, and/or text (sometimes referred to as a dashboard). One common way is to use GREPLAY. Version 9.2 offers the ODS Graphics Template Language (GTL). There are also occasions when it is necessary to overlay the output of one SAS/Graph procedure on top of another rather than having the output of multiple procedures occupy its own space. While this is easier in 9.2, with a little effort the same result can be achieved using 8.2 and 9.1.

The figure below is an example of a graphic produced using 9.1, showing show nine vertical bar charts overlaid on a map. Regional map boundaries are also overlaid over the state boundaries. This output was produced in two ways: as a PDF file using the ODS option STARTPAGE=NEVER; as a PNG file using the annotate facility and the IMAGE function.



The data depicted in the graphic above comprise the change in the value of a variable from 1996 to 2003 in nine different regions of the United States. The bar charts show the level of the variable in each of the two years together with the percent change. PROC GCHART was used to create the bar charts, while PROC GMAP created the underlying map. The entire program is included in the zip file on the Proceedings CD. Two different programs are shown, one for creating PDF output, the other for creating PNG output. There are some tasks that are common to the SAS coded used to create both output types. One of those tasks is to create the format used to group the states into nine regions.

A format, ST2RG (or 'state-to-region'), is created using a CNTLIN data set. First, a data set, ST, is created in which each state plus the District of Columbia is assigned a region. Data set ST contains all the variables necessary to create a format: FMTNAME, START, and LABEL. PROC FORMAT uses data set ST to create the numeric format ST2RG. The next step is to use the format to create the region overlay (thick boundaries) for the map as an annotate data set.

The US map data set is used (a SAS-supplied map data set). The format ST2RG adds a new variable to that data set, REGION. The data are sorted by the new variable and PROC GREMOVE is used to create another map data set, USGR, which contains only the regional boundaries with no internal state boundaries. The map data set USGR is then transformed into an annotate data set name REGION_OUTLINE. The variables HSYS and SIZE control the thickness of the region boundaries. With HSYS assigned a value of 3 and SIZE a value of 2, the thickness of region boundaries is 2% of the graphics output area.

The map for either PDF or PNG output is created using a PATTERN statement, PROC GMAP, and the region outlines from the annotate data set.

```
pattern v=me c=gray98;
proc gmap data=maps.us (obs=1) map=maps.us all anno=region_outline;
id state;
choro state / discrete nolegend coutline=gray98;
run;
quit;
```

The SAS-supplied US map data set is used as both the DATA and MAP data set. The use of only one observation from the DATA data set in combination with the ALL option is an easy way to create a map that shows only outlines, in this case those of states. The PATTERN statement in combination with the COUTLINE option in PROC GMAP specifies that the states be outlined in light gray. The region outlines in the annotate data set are added with the ANNO option.

The SAS code thus far is common to producing both PDF and PNG output. The remaining task of overlaying the bar charts requires a different approach for the PDF and PNG output. A vertical bar chart is created using PROC GCHART with a VBAR statement. Assume you have a data set TPAY that contains the following numeric variables: REGION (values 1 though 9); X (value in 1996 and 2003 to be shown in the bar chart); TIME (a variable with the value 1996 or 2003). To produce a bar chart for one region (New England) similar to that shown in figure x, the following statements can be used.

```
proc gchart data=tpay;
where region eq 1;
vbar time / type=mean discrete sumvar=x mean;
run;
quit;
```

Some customizing is needed to achieve the look of the bar charts in figure x. That is accomplished using AXIS statements.

```
axis1 label=('NEW ENGLAND'          j=c '%CHANGE 48.7') ;
axis2 label=('MIDDLE ATLANTIC'      j=c '%CHANGE 21.1') ;
axis3 label=('SOUTH ATLANTIC'       j=c '%CHANGE 18.7') ;
axis4 label=('WEST NORTH CENTRAL'   j=c '%CHANGE 63.3') ;
axis5 label=('EAST NORTH CENTRAL'   j=c '%CHANGE 33.3') ;
axis6 label=('EAST SOUTH CENTRAL'   j=c '%CHANGE 48.4') ;
axis7 label=('WEST SOUTH CENTRAL'   j=c '%CHANGE 22.5') ;
axis8 label=('MOUNTAIN'             j=c '%CHANGE 50.9') ;
axis9 label=('PACIFIC'              j=c '%CHANGE 23.9') ;

axis10 order=0 to 20 by 5 label=none value=none style=0 major=none minor=none;

proc gchart data=tpay;
where region eq 1;
vbar time / type=mean discrete sumvar=x mean raxis=axis10 maxis=axis1;
run;
quit;
```

An AXIS statement is written for each region that controls the text that appears on the midpoint axis (MAXIS option, or the x-axis at the bottom of the chart). A different AXIS statement is used for each region. A tenth AXIS statement is used for each chart to control the appearance of the response axis (RAXIS option, or y-axis at the left of each chart). On the response axis, all text is suppressed (LABEL and VALUE options) and no lines are drawn (STYLE,

MAJOR, and MINOR options).

Without any control of size or location, the above statements produce a bar chart that fills the entire graphics output area. Figure x requires that the bar charts are reduced in size and are placed in different locations on the map. Both the size and location of the charts is controlled by a GOPTIONS statement. The "New England" bar chart shown in figure x can be sized and drawn in the proper location if the following GOPTIONS statement is used prior to using PROC GCHART.

```
goptions hsize=2.5 in vsize=2.5 in horigin=8.75 in vorigin=6.75 in;
```

The bar chart will be drawn within a 2.5 inch square (HSIZE and VSIZE controlling horizontal and vertical size) with the lower left corner of that square being 8.75 inches from the left and 6.75 inches up from the bottom of the graphics output area (HORIGIN and VORIGIN controlling the horizontal and vertical location. All nine bar charts will be the same size, but the location varies. For an individual bar chart, the GOPTIONS statement can be placed inside the PROC GCHART code.

```
proc gchart data=tpay;
goptions horigin=8.75 in vorigin=6.75 in;
where region eq 1;
vbar time / type=mean discrete sumvar=x mean raxis=axis10 maxis=axis1;
run;
quit;
```

The only portion of the above SAS code that changes when creating a bar chart for another region is: the WHERE statement; the value of the MAXIS option; the values used for the HORIGIN and VORIGIN options. This is a good time to use a short macro to avoid having to repeat nine times the writing of slightly modified SAS code.

```
%macro bars(reg,h,v);
goptions horigin=&h in vorigin=&v in;
where region eq &reg;
vbar time / type=mean discrete sumvar=x raxis=axis10 maxis=axis&reg mean;
run;
%mend;
```

The values for the region, axis, and location are replaced by macro variables. The code to produce all nine charts in the proper locations is reduced to:

```
proc gchart data=tpay;
%bars(1,8.75,6.45) %bars(2,7.75,4.90) %bars(3,7.50,3.00)
%bars(4,4.00,5.00) %bars(5,6.00,4.25) %bars(6,6.10,2.50)
%bars(7,4.00,1.85) %bars(8,1.95,4.20) %bars(9,0.00,2.00)
quit;
```

The final detail is how to overlay the bar charts on top of the map. For the PDF version of the graphics output, this is accomplished using the ODS statement

```
ods pdf file='z:\map_bars.pdf' notoc startpage=never;
```

prior to drawing the map with PROC GMAP and the bar charts with PROC GCHART. The ODS statement directs graphics output to a PDF file. The NOTOC option suppresses the placing of a bookmark in the PDF file. The STARTPAGE option with a value of "never" requests that all graphics output produced prior to the closing of the PDF file be drawn on the same page. The only details that remain are handled with a GOPTIONS statement.

```
goptions reset=all ftext='Helvetica/bo' htext=3.5 gunit=pct ctext=blue lfactor=3;
```

Since the graphics output will be a PDF file, a bold-Helvetica font is used. The height of all text (unless otherwise specified) will be 3.5% of the graphics output area and will be blue. Since lines drawn in PDF files tend to be quite thin, the LFACTOR option is used to increase the width of all lines drawn with PROCs GMAP and GCHART. The entire SAS job with all the pieces in place is shown in the zip file on the Proceedings CD, and the approach taken to create the map shown above as a PDF file can be summarized as follows.

- #1 specify the 'look' of the text and lines with a GOPTIONS statement
- #2 use an ODS statement to direct graphics output to a PDF file
- #3 draw the map, annotated with region boundaries, using PROC GMAP
- #4 draw the bar charts using PROC GCHART

Though creating figure x as a PNG file has much SAS code in common with that use to create a PDF file, there are also a number of differences. Since there is no equivalent to STARTPAGE=NEVER for bit mapped graphics output, another method must be used. Rather than using PROC GCHART to create bar charts that are drawn on top of the map, PROC GCHART is used to create GIF files, or images, that are placed on top of the map using an annotate data set.

Creating one bar chart as a GIF file can be done with the following.

```
goptions reset=all device=gif gsfname=gout transparency xpixels=600 ypixels=800
lfactor=3 ftext="Arial/bo" htext=6 ctext=blue gunit=pct;
filename gout "z:\bar1.gif";

proc gchart data=tpay;
where region eq "1";
vbar time / type=mean mean discrete sumvar=x width=25 space=10;
run;
quit;
```

The GOPTIONS statement specifies the GIF device driver and indicates that graphics output for any subsequent procedures is to be directed to the FILEREf GOUT. The remaining options control the appearance of the bar chart. The most important of these is TRANSPARENCY. This provides a "see through" background for the bar charts so the map may be seen underneath. A FILENAME statement associates an external filename with the FILEREf GOUT. PROC GCHART is similar to that used when creating PDF output. The only additions are a WIDTH and SPACE option to control the appearance of the bars.

This code must be repeated nine times, once per region. As with the PDF file, this is a good time to use a macro to reduce the amount of SAS code to be used.

```
%macro bars(reg);
where region eq &reg;
filename gout "z:\bar&reg..gif";
vbar time / type=mean mean discrete sumvar=x
          raxis=axis10 maxis=axis&reg width=25 space=10;
run;
%mend;
```

The FILENAME statement is moved inside PROC GCHART and allows the output to be directed to a different file each time the macro is used. The one macro variable ® chooses the region (WHERE), directs the output (FILENAME) and selects the correct AXIS statement (MAXIS). The AXIS statements are the same as those used to create the PDF output.

```
proc gchart data=tpay;
%bars(1) %bars(2) %bars(3) %bars(4) %bars(5) %bars(6) %bars(7) %bars(8) %bars(9)
quit;
```

The next step is to use the GIF files in an annotate data set. We can add one image (the GIF file with the bar chart for region 1) to an annotate data set as follows.

```
%let xsize=14;
%let ysize=28;

data annobars;
length function $8 imgpath $30;
retain xsys ysys '3' when 'a' style 'fit';
```

```

imgpath="z:\bar1.gif";
function='move'; x=84-.5*&xsize; y=80-.5*&ysize; output;
function='image'; x=x+&xsize; y=y+&ysize; output;
run;

```

The macro variables &XSIZE and &YSIZE allow you to 'experiment' with how the bar charts appear without having to modify subsequent SAS code that creates the annotate data set ANNOBARS. Understanding all the abilities of an annotate data set is beyond the scope of this paper. For now, it is enough to understand that placing an image on top of graphics output requires a number of options: WHEN is specified as after (A) the procedure output is created; STYLE specifies that the image be fit into the assigned space rather than tiled; IMGPATH specifies the external file that contains the image. Setting XYSY and YSYS both with a value of 3 means that any values involved in setting the position of the image are expressed in percentage of graphics output area (remember, with the PDF out, the bar charts were positioned using inches, not percentages. Once again, this operation must be repeated none times, so yes, another macro is used.

```

%macro image(reg,x,y);
function='move'; x=&x-.5*&xsize; y=&y-.5*&ysize; output;
imgpath="z:\bar&reg.gif";
function='image'; x=x+&xsize; y=y+&ysize; output;
%mend;

data addbars;
length function $8 imgpath $30;
retain xsys ysys '3' when 'a' style 'fit';
%image(1,84,80) %image(2,75,67) %image(3,74,42)
%image(4,50,65) %image(5,63,58) %image(6,64,39)
%image(7,47,35) %image(8,32,60) %image(9,13,46)
run;

```

The final step is to draw the map with bars added using the annotate data set. Notice that two annotate data sets are used, one with the region outlines and the other with the bar charts.

```

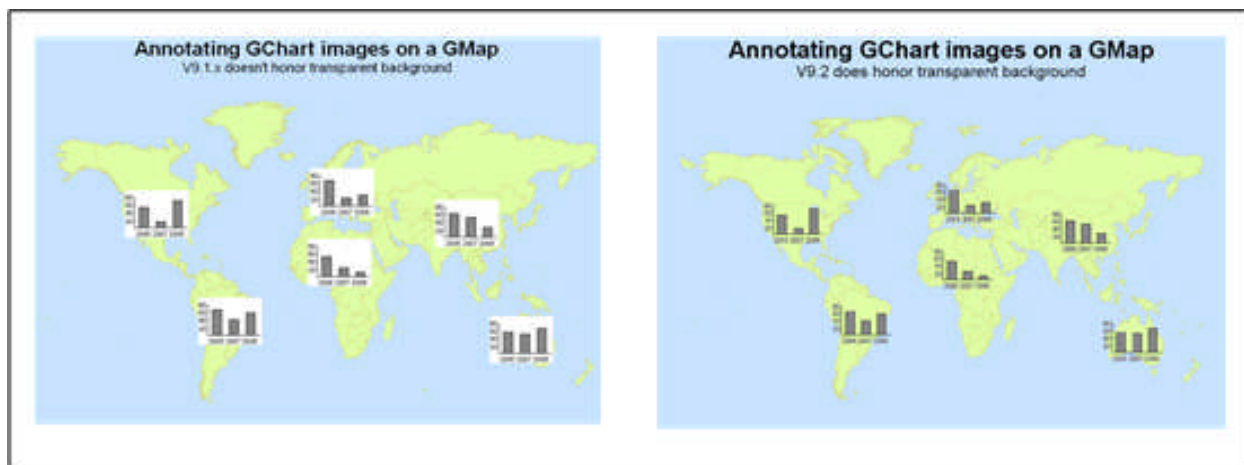
goptions device=png xpixels=2500 ypixels=1800;
pattern1 v=me c=gray88;

/* Add some space on the top, left, and right */
title1 ls=5;
title2 a= 90 ls=8;
title3 a=-90 ls=8;

filename gout 'z:\map_bars.png';
proc gmap data=us (obs=1) map=us all anno=region_outline;
id state;
choro state / discrete nolegend coutline=gray98 anno=addbars;
run;
quit;

```

Both of these methods (PDF and PNG) work with this particular example in SAS version 9.1. However, in 9.1, there are some problems when annotating SAS/GRAPH output with the IMAGE function. Sometimes the transparency option of the original image is ignored and annotated images appear with solid backgrounds. Also, images are sometimes a bit grainy. Both of those problems are fixed in 9.2.



Both the example above and the example below (look familiar?) demonstrate the ability to use annotated images with transparent backgrounds in 9.2. In 9.1, you could use the 'image' function to annotate an image (such as the image of a bar chart) onto a GMAP. Even if the image had a transparent background, that transparency was not honored by GMAP and ANNOTATE, and the background of the annotated image showed up as a solid rectangle, obscuring the map.

```
data anno_plots; set anno_plots;
  length function style $8 html $1024;
  retain xsys ysys '2' hsys '3' when 'a';
html=
'title='||quote(trim(left(name)))|| ' ' || ' ' || ' ' ||
'href="http://www.google.com/search?q=data+about+'||'+ '||trim(left(lowercase(name)))||
'";
  function='move'; output;
  xsizer=500;
  ysize=375;
  x=x-xsizer; y=y-ysize; output;
  imgpath=trim(left(plotname))||'.gif'; style='fit';
  function='image'; x=x+(xsizer*2); y=y+(ysize*2); output;
run;
```

In 9.2, transparency in annotated images is honored. Therefore, when you annotate them on a GMAP, the rectangular background of the image does not obscure the map. This really helps your annotated images blend in with the GMAP and you can achieve some very nice special effects with this capability! Note: in creating the plots (bar charts) used to annotate the GMAP, use GOPTIONS TRANSPARENCY as shown below.

```
GOPTIONS DEVICE=gif;
goptions transparency;

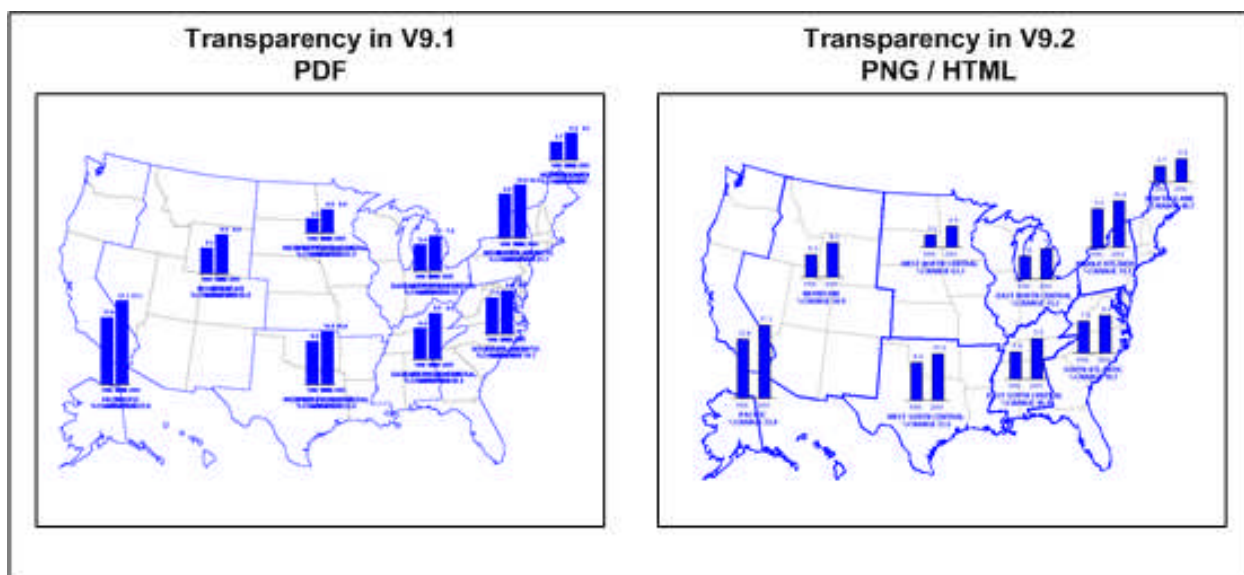
ODS LISTING CLOSE;
ODS HTML path=odsout body="&name..htm" (title="Plots");
goptions noborder;

goptions xpixels=100 ypixels=75;
goptions cback=white;
goptions gunit=pct htitle=1 htext=15 ftitle="arial" ftext="arial";

axis1 label=none minor=none order=(0 to 80 by 20) offset=(0,0) value=(t=1 ' ');
axis2 label=none minor=none offset=(15,15);
pattern1 v=s c=gray88;
```

```
proc gchart data=my_data (where=(continent='North America'));
  vbar year / discrete
  type=sum sumvar=value
  width=15 space=12 coutline=black
  raxis=axis1 maxis=axis2 noframe
  name="na_plot";
run;
. . .
```

This same technique can be used to create the 9.1 PDF/PNG transparency map shown above and below in a PNG map in 9.2 (shown below.) First GIF bar charts are created with transparent backgrounds using GOPTIONS TRANSPARENCY, and then those bar charts are annotated on a GMAP.



Another 9.2 PNG map example (below) is based on one of Robert Allison's samples. This sample incorporates a number of the techniques described above, and more. Unlike the examples above which both use annotate to produce the overlay effect, this example uses GREPLAY instead. It also utilizes a relatively obscure SAS-provided dataset, SASHELP.MAPFMTS, to produce current world country names matching the world map data set. Full code for the sample is available on the Proceedings CD: a salient "snippet" follows.

To use one of the formats for maps, you must specify the SASHELP.MAPFMTS catalog on the FMTSEARCH= option on a SAS OPTIONS statement:

```
options fmtsearch=(sashelp.mapfmts);

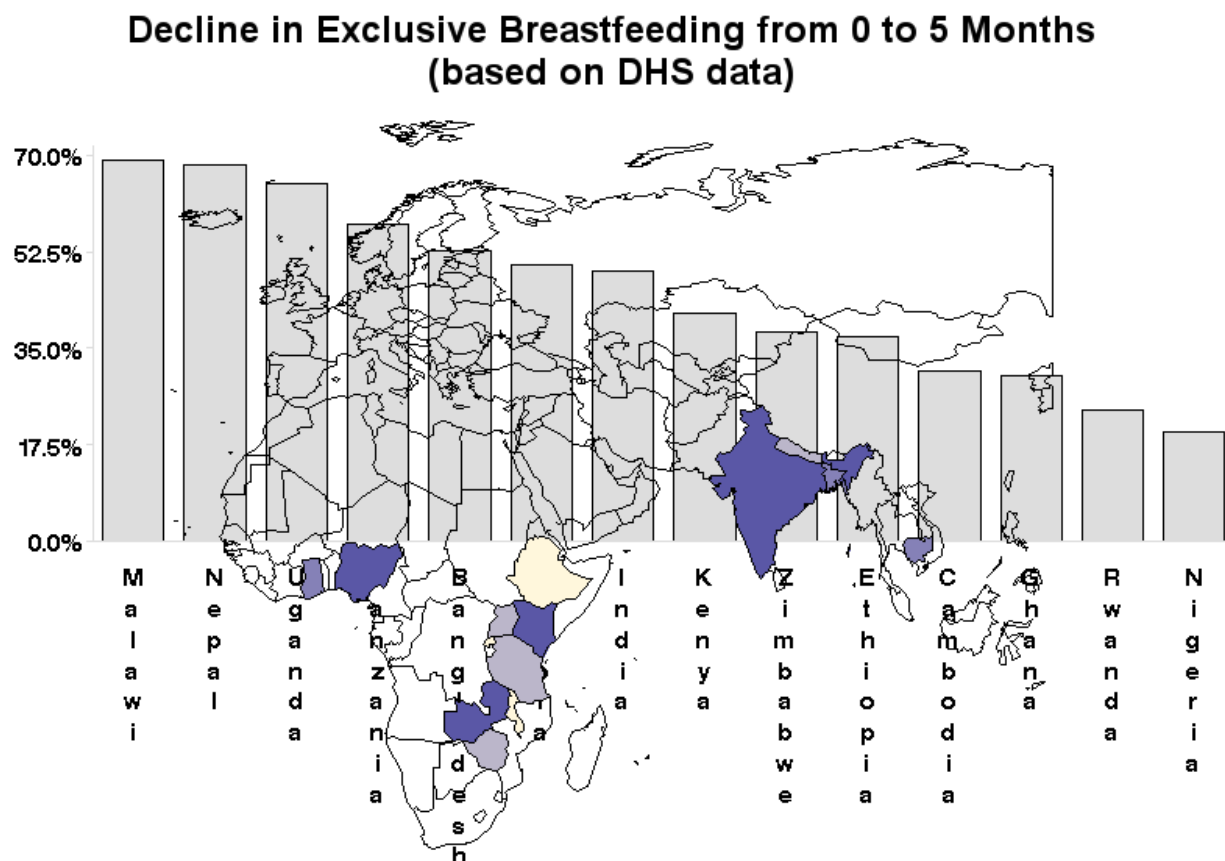
data work.world;
  set work.world;
  length country_name $20;
  country_name=put(country,glcnsml.);
  country_name=propcase(country_name);
run;
```

For more information on SASHELP.MAPFMTS, see

<http://support.sas.com/documentation/cdl/en/graphref/59607/HTML/default/a002230502.htm>

Other features of the program are use of PROC TEMPLATE to define colors used in the GMAP procedure as discussed above, reduction of the map area and density using PROC SQL; a PROC GMAP (using nodisplay) to create the underlying map; a PROC GCHART to create the overlay bar chart (again using nodisplay), and a PROC GREPLAY to display both images in the same space. The HTML file accompanying the PNG file has additional

functionality. Mousing over the colored countries provides additional information on breastfeeding rates and GNI in a tooltip for each country.



CONCLUSION

SAS/GRAPH and ODS provide SAS programmers with tools to create attractive and informative maps. Among these tools are: SAS-supplied map data sets; PROC MAPIMPORT which enables use of external map data sets; functions and procedures; helpful utility files; and numerous samples including macros, programs and data. SAS 9.2 facilitates the creation of maps that make people say "Wow! You made that map with SAS/GRAPH?"

REFERENCES & RECOMMENDED READING

<http://support.sas.com/documentation/onlinedoc/index.html>

<http://support.sas.com/rnd/datavisualization/maponline/html/home.html>

<http://support.sas.com/rnd/papers>

<http://support.sas.com/samples>

<http://ftp.sas.com/techsup/download/sample/graph/other-color.html>

http://arcdata.esri.com/data/tiger2000/tiger_download.cfm

<http://www.census.gov/geo/www/cob/index.html>

<http://www.cdc.gov/epiinfo/shape.htm>

<http://research.cip.cgiar.org/gis/index.php>

<http://support.sas.com/techsup/technote/ts688/ts688.html>

<http://www.colorbrewer.org>

<http://support.sas.com/documentation/cdl/en/graphref/59607/HTML/default/a002230502.htm>

<http://robslink.com/SAS/Home.htm>

Zdeb, Mike and Allison, Robert. "Stretching the Bounds of SAS/GRAPH Software." *Proceedings of the Thirtieth Annual SAS Users Group International Conference*. April 2005.

Zdeb, Mike and Hadden, Louise. "Zip Code 411: A Well Kept SAS Secret." *Proceedings of the Thirty-First Annual SAS Users Group International Conference*. March 2006.

Zdeb, Mike. 2002. *Maps Made Easy Using SAS*. Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS & DISCLAIMERS

Darrell Massengill and Liz Simon of SAS who work tirelessly to improve and facilitate the use of SAS/GRAPH and mapping with SAS and maintain the MAPSONLINE site.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. indicates USA registration. Other brand and product names are registered trademarks or trademarks of their respective companies.



CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Robert Allison
robert.allison@sas.com

Louise Hadden
louise_hadden@abtassoc.com

Mike Zdeb
msz03@albany.edu

Sample code for the published examples is available from the authors upon request, and in a zip file in the NESUG 2008 Proceedings. These samples are being provided "AS IS" without any express or implied warranties. SAS Institute Inc. is not liable for any damages caused by the use of this software.