

Working With PROC GMAP and a Note About Census ZCTAs

Deron Ferguson, WA State Office of Financial Management, Olympia, WA
Presented at the Pacific Northwest SAS Users Group Conference, Seattle, WA
September 17-18, 2007

Abstract The SAS/GRAPH GMAP procedure is a versatile tool for making maps in SAS. This paper describes a work-around for a known bug when importing Census 2000 ZCTA boundaries with MAPIMPORT, and on the way it illustrates a few basics of GMAP and related mapping procedures.

Introduction

Like pictures, maps can be worth a thousand words. Maps are often very useful in research reports, slides or other documents. Fortunately, it is a fairly simple matter to produce basic thematic maps for your reports by using GMAP. There are a number of good primers for getting started with GMAP and its related mapping procedures. Mike Zdeb's excellent book, *Maps Made Easy Using SAS*, provides a clear and easy to follow introduction to mapping with GMAP. Several SUG papers are also listed at the end of this paper, and many more are available on the SAS website.

In this paper I document a work-around for a bug with respect to importing Census 2000 Zip Code Tabulation Area (ZCTA) boundaries. The solution for this bug was provided by Bob Allison, who responded to my inquiry on SAS-L. In the process of describing the bug and its fix, I use and explain several of the SAS/GRAPH mapping procedures, which may be useful to those who are new mapping in SAS.

The Census' ZCTA boundaries were developed for zip-code-like reporting of census data. Because they were developed specifically for reporting Census data, they do not exactly replicate zip code boundaries available from private vendors with which many users may be familiar. The ZCTAs were defined by aggregating Census blocks with the same predominant zip code found in the Master Address File (MAF). I have found that in urbanized areas, the zip code boundaries are not usually very different from those of private vendors, however, in less populated territory the boundaries can diverge considerably. More information about ZCTAs can be found on the web at the URL's noted at the end of this paper. It should be remembered that the USPS itself does not consider zip code areas as having precise "boundaries." They use zip codes only for delivery route identification and planning, and the actual territory related to zip codes is subject to constant change in support of the goal of delivering mail, *not* of reporting data or area statistics.

Importing Map Boundaries

The MAPIMPORT proc enables us to import any ESRI shapefile (.shp), which is the native format for boundary files used by ESRI's ArcGIS software, and which is probably the most common format found in the GIS industry. Governments at all levels nowadays routinely provide boundary files on the web in .shp format for public use, such as the Census Bureau, departments of transportation, municipal engineering departments, etc.

If you use GIS software, you can of course customize your boundary files before importing them to SAS. For example, you may wish to edit some boundaries, group areas into

regions (although SAS/GRAPH can also do this), or perform any number of more complicated GIS tasks to produce shapefiles for use in SAS/GRAPH.

To begin, we use MAPIMPORT to import the ZCTA 5-digit boundary layer obtained from the Census Bureau's Cartographic Boundary Files website, located at the URL noted at the end of this paper. This site provides all of the geographic boundaries used in the 1990 and 2000 censuses.

```
libname shp "C:\mapdata\shp"; * Library for map work, location of shapefile;

/* Import ZCTA5 boundaries */
proc mapimport contents
    out= shp.wa_zcta5_unprj
    datafile= "C:\mapdata\shp\zt53_d00.shp" ;
run;
```

The contents option produces the resulting listing of the shapefile's fields and their types, as shown in the log, below.

```
Contents of the Input file

Input file name: C:\mapdata\shp\zt53_d00.shp

List of Fields and Attributes

# Field          Type Width Decimals
1 AREA           NUM    20    5
2 PERIMETER      NUM    20    5
3 ZT53_D00_      NUM    11    0
4 ZT53_D00_I     NUM    11    0
5 ZCTA           CHAR     5    0
6 NAME           CHAR    90    0
7 LSAD           CHAR     2    0
8 LSAD_TRANS     CHAR    50    0
10 run;

Number of Fields:      8
Number of Records:    962
NOTE: The data set SHP.WA_ZCTA5_UNPRJ has 147159 observations and 12 variables.
NOTE: PROCEDURE MAPIMPORT used (Total process time):
      real time          5.75 seconds
      cpu time           0.46 seconds
```

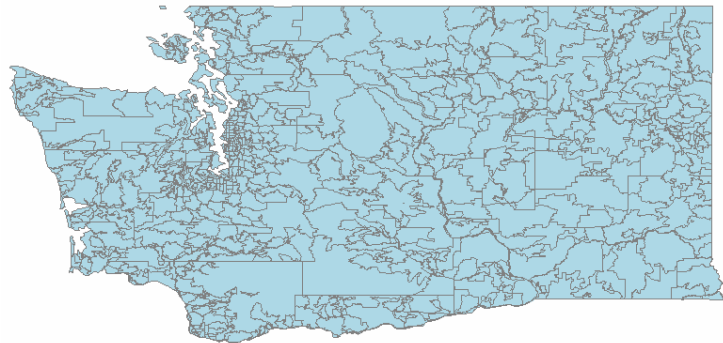
The fields that will be of interest to us are the ZT53_D00_I and ZCTA fields. The ZT53_D00_I field is merely a by-product of the cartographic file generation at the Census Bureau, but it happens to come in handy for our bug fix, as discussed below. The ZCTA field identifies the 5-digit zip code area. Some areas have their two last digits truncated and replaced with either "XX" or "HH." The XX code indicates large, undeveloped territory where there were no addresses assigned in the Master Address File (MAF). The HH code is assigned to large water areas, such as bays, rivers, coastline, etc. The AREA and PERIMETER fields should not be used.

At this point, we can try a test run of the ZCTA boundary file to see how it looks using proc GMAP:

```
pattern1 c=lightblue v=msolid r=9999;  
proc gmap  
  map= shp.wa_zcta5_unprj  
  data= shp.wa_zcta5_unprj;  
  id name;  
  choro name / coutline = gray nolegend;  
run;  
quit;
```

The resulting map is shown in Figure 1. In the GMAP procedure, the map= option refers to the map dataset to be used (the one we just imported), and the data= option points to the response dataset holding the data that we would like to map, such as population counts, growth rates, etc. In this case, I am pointing to the map dataset itself for the “response data,” as we just want to view the boundaries for now. Later on, we will point to a separate dataset containing the response data. The ID statement identifies the field that uniquely identifies each area, which in this case is the ZCTA, the 5-digit zip code tabulation area. This does not necessarily imply that it identifies each polygon – a single area coded in the ID field may be made up of several separate polygons, for example, many of Washington’s ZCTAs include islands in the Puget Sound, or multiple areas. Finally, the choro option specifies the type of map we want: a choropleth map. A choropleth map displays areas with shading according to values in the response dataset.

Figure 1. Imported ZCTA5 Boundaries



The pattern statement establishes the symbolization for each area: light blue, solid fill, to be repeated 9,999 times. Without this statement, SAS would select a unique pattern/color combination for each zip code area. Because we don’t know off-hand how many ZCTAs there are in Washington, we just specify a high number of areas to repeat this pattern, or 9,999 times. Another way to get a single color would have been to specify a constant in the choro statement, such as the LSAD field (Legal/Statistical Area Description), which happens to be the same for every record in this map dataset (with a value of “Z5” – a code for “5-digit ZCTA”). In that case, only one pattern would be used for every area, and no repeat option would be needed.

In addition to choropleth maps, the GMAP procedure can produce three other types of maps: prism, block, and surface. These are described in the GMAP documentation and some of the references at the end of this paper.

Map Projection

You may notice that the map in Figure 1 looks “flat,” compared to the depiction of Washington you may be used to seeing. This is because the map is unprojected. Projection is the process of translating points on a 3-dimensional sphere to a 2-dimensional plane.

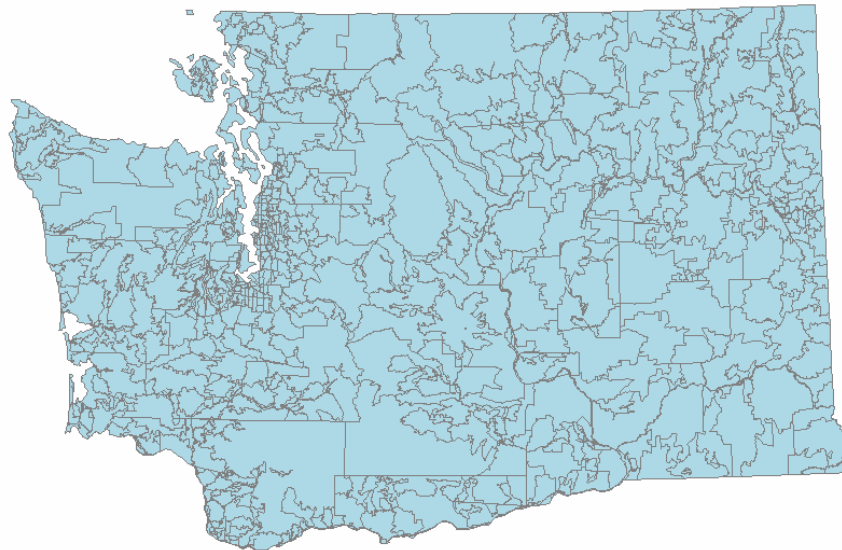
Some distortion is inevitably involved in this process. Think of trying to cut a basketball and spread it flat on a piece of paper – some distortion of direction and/or area size of the surface of the basketball must occur. All of the Census boundary products are unprojected, meaning that coordinates are given in degrees of latitude and longitude. When mapped as in Figure 1, this will make upper-latitude areas “look” flat because we are used to seeing the projected depictions of these areas that portray these areas’ actual size.

To project our map, we can use GPROJECT, and we’ll use the Albers equal-area projection, resulting in the map areas’ relative size being more accurately represented:

```
proc gproject data=shp.wa_zcta5_unprj
    out=shp.wa_zcta5_prj
    project= Albers /* default, but indicate anyway */
    degree /* specifies that coordinates are in decimal degrees */
    eastlong ; /* specifies that longitude increases to the east */
    id zcta;
run;
quit;
```

Then, after running the same GMAP proc above but with the new dataset wa_zcta5_prj, Figure 2 is produced.

Figure 2. Projected ZCTA Boundaries



Fixing the ZCTA Bug

At first glance, the map in Figure 2 appears acceptable, however, if we look carefully we will find flaws in the drawing of some of the boundaries. To bring out the flaws more clearly, we can map the ZCTA3 boundaries. These are boundaries defined by only the first 3 digits of the 5 digit zip code. First, we alter the data set accordingly in order to map these areas, as well as add a flag to identify water areas (water areas are identified in the ZCTA file by records with “HH” as the last two characters in the ZCTA name):

```

data shp.wa_zcta5_unprj;
  set shp.wa_zcta5_unprj;
  water = 0; * Create a water flag;
  if substr(zcta,4,2) = 'HH' then water = 1; * Assign HH ZCTAs to water;
  ZCTA3 = substr(zcta,1,3); * Create 3-digit ZCTA field;
run;

```

Now we can draw a map of 3-digit ZCTAs, this time specifying new goptions (setting units to the percent of the graphic space and font to Arial), specifying options for the title and legend, and allowing GMAP to fill the areas and make a legend (note: some pattern statements for colorizing the map were used but are omitted here, to be illustrated later):

```

goptions reset=all;

goptions gunit=pct ftext='Arial' ; * Set units to percent and use Arial font;

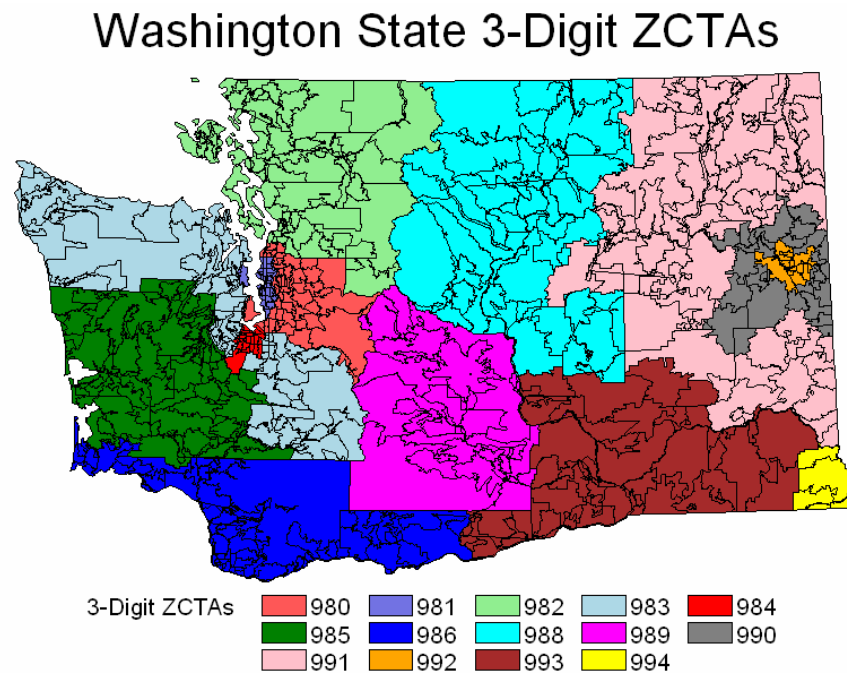
legend1 label=(height=4) shape=bar(5,3) value=(height=4); * Define legend;

title1 h=10 'Washington State 3-Digit ZCTAs';

proc gmap
  map= shp.wa_zcta5_prj
  data= shp.wa_zcta5_prj;
  id name;
  choro zcta3 / coutline = black legend=legend1 ; * Note ref to legend1;
  label zcta3 = '3-Digit ZCTAs'; * Specify clearer label for legend;
run;
quit;

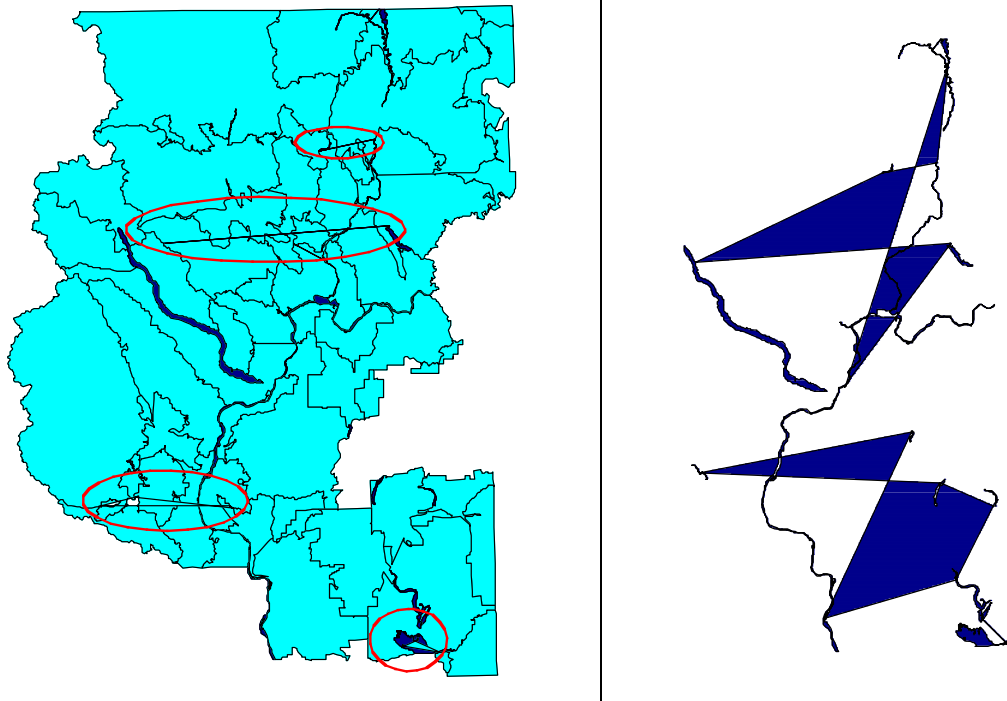
```

Figure 3. 3-Digit ZCTA Boundaries



In Figure 4, The 988 ZCTA3 area presents some of the easier to spot flaws (circled in red) in the drawing of the boundaries. These are mainly associated with drawing water polygons (dark blue areas). When we restrict the map to draw only the water bodies, the problem becomes even clearer (right side of Figure 4), although the flaws manifest a little differently.

Figure 4. Drawing Flaws in the 988 ZCTA3 Area



Essentially, this happens because the “segment” field created when the map was imported with MAPIMPORT was not created properly, resulting in SAS not knowing that it should pick up the pen and move to a different location to begin drawing a new polygon. The result is that a string of polygons is drawn with connecting lines – that is, they are drawn as one polygon. In order for SAS to know when to pick up the pen and begin drawing a new polygon – even though that polygon may still be part of the same area, such as “988HH” – the value of the segment field must change with each new polygon to be drawn.

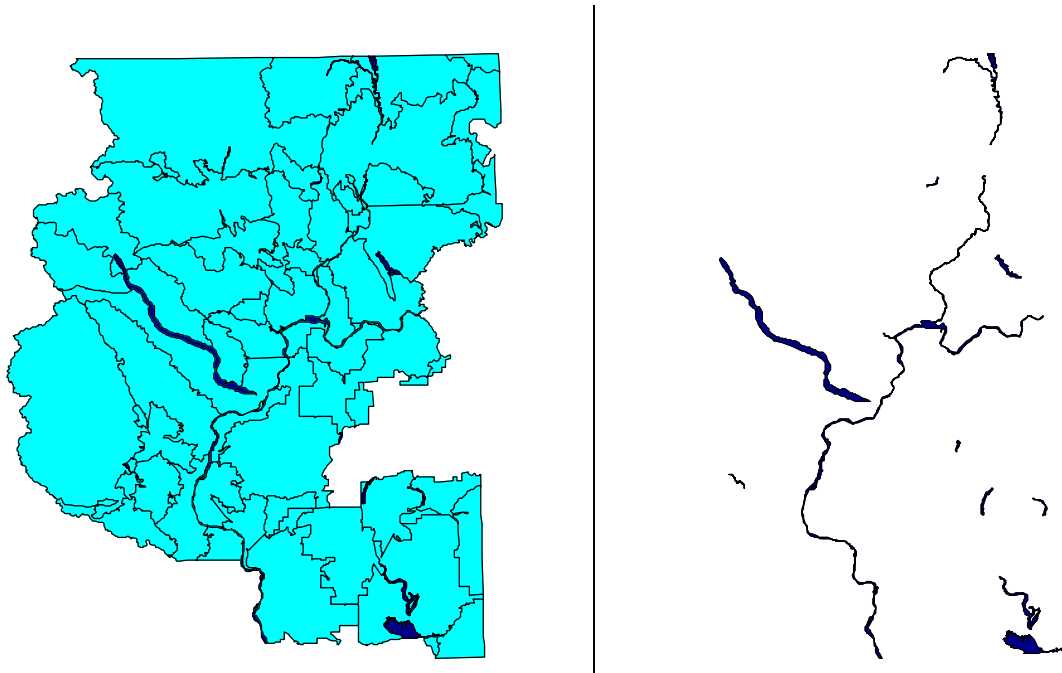
To fix this problem, the polygon ID field in the original Census ZCTA boundary file comes in handy. That field, ZT53_D00_I, identifies each individual polygon, and we can therefore use it to recode the segment field so that each polygon is uniquely identified and SAS consequently knows when to pick the pen up off the paper to begin drawing a new polygon.

```
data shp.wa_zcta5_prj2;
    set shp.wa_zcta5_prj;
    segment = (100*zt53_d00_I)+segment ;
run;
```

There are many ways to recode the segment field, but the above method has the advantage of preserving the original segment number in the last two digits of the recoded field, in case there may be some need to know that later. This assumes, probably safely, that values of the original segment field would never be above 99 (no individual ZCTA will have more than

99 polygons associated with it). As shown in Figure 5, the map is drawn correctly with the segment field recoded, with no connecting lines.

Figure 5. Corrected Drawing of the 988 ZCTA3 Area



Combining Map Layers

Now that we have a corrected version of the ZCTA map dataset to work with, we can improve upon it by adding a layer depicting Washington's shoreline and islands. We begin by importing and drawing the unprojected "major shorelines" boundary shapefile from the WA State Department of Transportation's website, located at the URL noted at the end of this paper.

```
proc mapimport contents
    out= shp.major_shore_unprj
    datafile= "C:\mapdata\shp\coast.shp" ;
run;
```

Contents of the Input file

Input file name: C:\mapdata\shp\coast.shp
List of Fields and Attributes

#	Field	Type	Width	Decimals
1	AREA	NUM	19	5
2	PERIMETER	NUM	19	5
3	COAST_	NUM	11	0
4	COAST_ID	NUM	11	0
5	WCODE	NUM	8	0

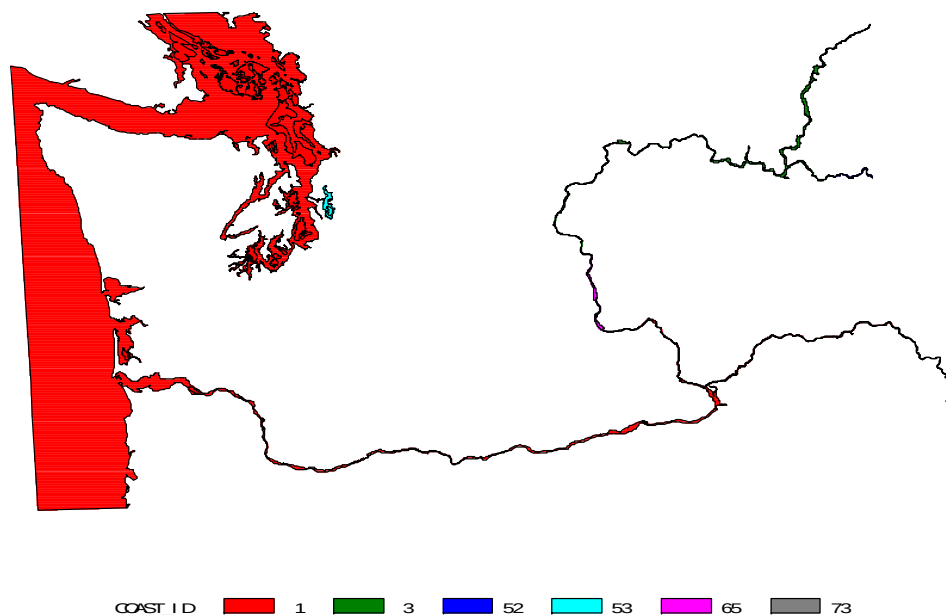
```

goptions reset=all;
proc gmap
  map= shp.major_shore_unprj
  data= shp.major_shore_unprj ;
  id coast_id;
  choro coast_id / discrete ;
run;
quit;

```

As with the ZCTA shapefile, the attribute fields for this shapefile include two polygon identification fields which were by-products of generating the layer in the GIS. It also includes a field called WCODE, which according the metadata posted on the website is supposed to flag whether a polygon is an island or water, however, every value in this column is "0." The boundaries as imported are mapped in Figure 6.

Figure 6. Washington Shoreline as Imported



Since we already have inland water bodies identified in the ZCTA boundary dataset, all we need from the shoreline dataset is the coastline (with the Pacific Ocean and Puget Sound, Coast_ID=1) and perhaps Lake Washington (Coast_ID=53). While the island polygons are specified in the shapefile (by unique segment numbers), they are not identified by Coast_ID. The following code creates a new coast boundary dataset, corrects the WCODE field so that it identifies whether a polygon is water or land, and creates a unique id for each polygon, poly_id, by concatenating coast_id and the segment number.

You may also notice in Figure 6 that the coastline boundary extends well southward below the Columbia River into Oregon's coastline. For better appearance, we would prefer to have the south end of the coastline boundary to be a little further north, or more even with the Columbia River. This can be done fairly simply by flagging the points we don't want to be drawn. In the code below, I have flagged the points west of longitude 123W and south of latitude 45.97N, which I determined by consulting a state map and a little trial and error to

refine my initial guess of 46N to a little further south (45.97N). Note that this is easier done *before* projecting the map, as the X and Y fields are degrees longitude and latitude, respectively, and therefore much easier to use for modifying boundaries.

```
data shp.coast_unprj;
  set shp.major_shore_unprj;
  if coast_id in (1,53); * Select only coastline and Lake Washington;

  * Create unique ID for each polygon;
  poly_id = compress(coast_id || '_' || segment);

  * Recode wcode flag for islands;
  if coast_id = 1 and segment > 1 then wcode = 1;

  cut = 0; * Create flag for cutting bottom of coastline;
  if poly_id='1_1' and x < -123 and y < 45.97 then cut = 1; * Define cut;
run;
```

After adjustments to the coast boundaries, we can draw the map. Note that by using the cut flag created above as a where= clause in the map statement, we exclude the southern part of the coastline that we do not want to have drawn. Also, by using a flag instead of eliminating the points altogether, we have preserved the points incase we want to use them for some other application.

```
goptions reset=all;
pattern1 c=lightblue ;
pattern2 c=white;
proc gmap
  map= shp.coast_unprj (where=(cut=0))
  data= shp.coast_unprj ;
  id poly_id;
  choro wcode / coutline=gray discrete ;
run;
quit;
```

Figure 7. Adjusted Shoreline Boundaries



Now we are ready to combine the ZCTA and coastline datasets. After applying the correction to the segment field in the unprojected ZCTA file, we combine the two datasets with a data step, with a few modifications along the way.

```
/* Apply correction to segment field of unprojected ZCTA */
data shp.wa_zcta5_unprj_adj;
    set shp.wa_zcta5_unprj;
    segment = (100*zt53_d00_I)+segment ;
    zip = zcta; * Duplicate field for matching later;
run;
/* Combine water with zips before projecting */
data shp.combined_unprj ;
    set shp.coast_unprj (in=shore
        keep = x y segment wcode cut where=(cut=0) ) /* Exclude south part*/
        shp.wa_zcta5_unprj_adj (in=zcta
        keep = x y segment name zip _id_ ) ;

    if shore = 1 then do;
        if wcode = 0 then do; name = '00000'; zip = '00000'; _id_ = 1; end;
        else if wcode = 1 then do; name = '00001'; zip = '00001'; _id_ = 1; end;
    end;

    if substr(zip,4,2) = 'HH' then do;
        name = '00000'; zip = '00000'; _id_ = 1; end;
    drop wcode;
run;
```

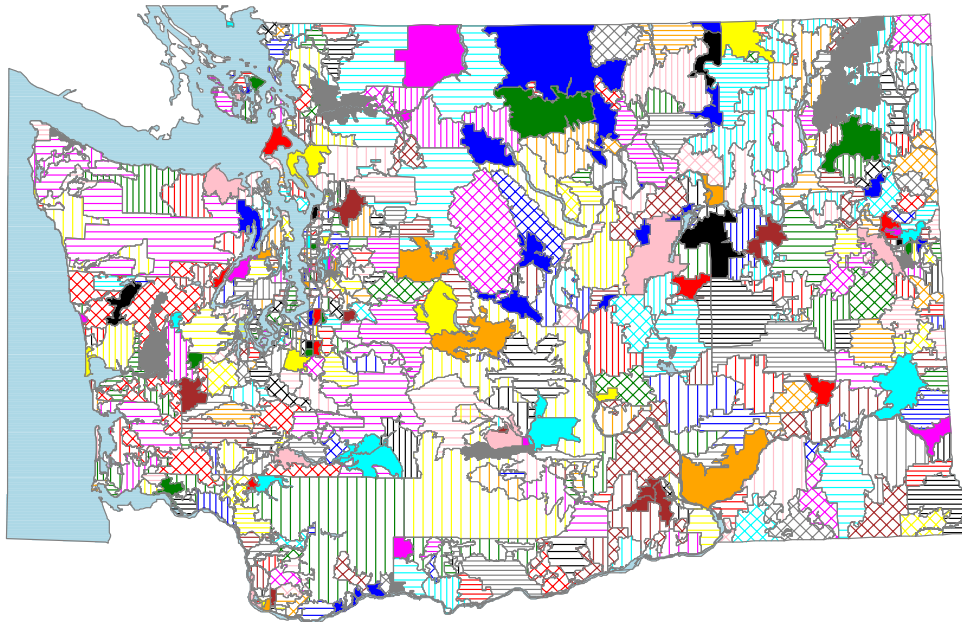
The modifications include the following: First, to be able to shade the shoreline areas and islands separately, they will need to have be uniquely identified by the variable used in the choro statement, which in this case is zip. The water areas are recoded to zip = 00000 and the islands to zip = 00001. This should not present any problem for joining datasets keyed on Washington zip codes to the map dataset. Second, the ZCTAs truncated with 'HH' are recoded to zip = 00000. These will have the same color as the coastline.

Note that in combining the coast and ZCTA map datasets, the coast was set first. The order is important: if we think about boundary datasets as layers, we want the coast to be drawn first, and then the ZCTAs to be drawn over that. The areas that come later in the data set (i.e., as observation number increases) are drawn last, and so will overwrite any areas drawn earlier.

After projecting the combined dataset, submitting the code below will draw the map. Note that pattern1 defines colors for the water areas (zip=0000) and pattern2 defines the islands (zip=00001). SAS assigns default colors and patterns to the remaining areas, as they come in order.

```
goptions reset=all;
pattern1 c=lightblue ;
pattern2 c=white ;
proc gmap
    map= shp.combined_prj
    data=shp.combined_prj all;
    id name ;
    choro zip / coutline=gray cempty = white discrete ;
run;
quit;
```

Figure 8. Combined Coast and ZCTA Boundaries



Mapping Response Data

The combined dataset is now ready for mapping response data. As a simple demonstration, we can produce a choropleth map of Census 2000 population by Washington ZCTA. One natural advantage of using the ZCTAs is that undeveloped areas (with 'XX' in the ZCTA code) are left blank, providing a more realistic picture of where people actually live. Vendor-provided zip code boundaries normally extend into mountainous regions and other unpopulated territory, suggesting (at least subjectively) that the human activity being mapped extends evenly across such territory.

The population dataset has three fields: ZIP, POP, and NUM_BLOCKS. These are the ZCTA codes, the 2000 population, and the number of blocks in the ZCTA, respectively. Before mapping the population data, we must add two records with zip=00000 and zip=00001 to the dataset so that they will be picked up in the choro statement. We create the records and add them in the subsequent data step, where we also categorize population into 5 categories, plus two categories for the water and island zip codes. The FORMAT proc creates a format for use in producing the map.

```
libname mapdata 'C:\mapdata';

data pop_header;
  input zip $5. pop num_blocks;
  datalines;
00000 0 0
00001 0 0
run;
```

```

data pop_data;
  set pop_header
    hosp.wa_zip2000_population;
  if zip = '00000' then pop_cat = 0;
  if zip = '00001' then pop_cat = 1;
  if 1 <= pop < 1000 then pop_cat = 2;
  if 1000 <= pop < 10000 then pop_cat = 3;
  if 10000 <= pop < 20000 then pop_cat = 4;
  if 20000 <= pop < 30000 then pop_cat = 5;
  if 30000 <= pop then pop_cat = 6;
run;

proc format;
  value pop_cat
    0 = 'Water'    1 = 'Not populated'  2 = 'Less than 1,000'
    3 = '1,000 - 10,000'  4 = '10,000 - 20,000'
    5 = '20,000 - 30,000'  6 = '30,000 or More';
run;

```

After these preliminary steps, the map can be produced. Patterns are defined for the 6 population categories (including the two “zero” categories, water and islands/unpopulated areas). Titles are defined, and the map graphic is output to a jpeg file with a combination of a file reference and specifying the jpeg graphic device using goptions. Note also that graphics can also be copied and pasted directly from the interactive SAS graphics viewer into documents in the Windows OS, but outputting them to files is obviously handy for later use.

```

goptions reset=all;

goptions gunit=pct ftext='Arial';

pattern1 c=lightblue ; pattern2 c=white ;
pattern3 c=lightyellow ; pattern4 c=lightgreen;
pattern5 c=green;
pattern6 c=red; pattern7 c=darkred;

title1 h=8 'Washington State Population, 2000';
title2 h=4 'Zip Code Tabulation Areas (ZCTAs)';
legend1 label=(height=4) shape=bar(4,3) value=(height=3); * Define legend;

filename gout 'C:\mapdata\wazcta_pop2000.jpeg';
goptions device=jpeg gsfname=gout xpixels=800 ypixels=600;

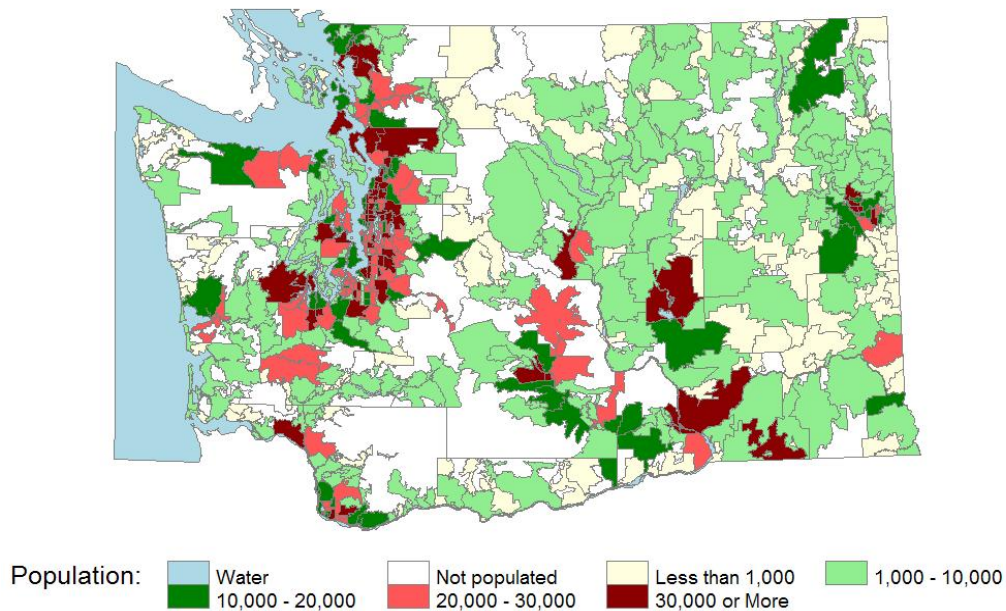
proc gmap
  map= shp.combined_prj all
  data=pop_data ;
  id zip ;
  choro pop_cat / coutline=gray legend = legend1;
  format pop_cat pop_cat.;
  label pop_cat = 'Population: ';
run;
quit;

```

Figure 8. Population Map

Washington State Population, 2000

Zip Code Tabulation Areas (ZCTAs)



Concluding Remarks

The goal of this paper was to document the simple fix for importing and using Census ZCTA boundaries for use with the GMAP procedure. While the Census 2000 ZCTA boundaries are now more than 7 years old, they are still useful for bridging alternative vintages of vendor-supplied zip codes or crosswalking data referencing zip codes from different periods. Obviously, a wealth of useful socioeconomic data produced by the Census Bureau is available for this geography, too.

Along the way some basic mapping steps were illustrated. One advantage of GMAP that was not illustrated is that once map boundaries are developed, they can be used in the automation the production of reports with the use of macros. Effective automation for web publication is also a possibility.

SAS/GRAPH offers many functions and a great amount of mapping flexibility. In addition to what was demonstrated here, maps can be drawn to scale, clipped, created with insets, annotated, and output with drill down capabilities, just to name a few possibilities. There is a wealth of information available for using GMAP. A sample of SUG papers and other resources are listed at the end of this paper.

Acknowledgements

Thanks goes to Robert Allison (SAS Institute, Inc.) for responding to my initial question on SAS-L about importing the ZCTA 2000 boundaries and sharing his code for the bug fix documented here.

Sample Papers

Introduction to GMAP by Jeffery Gilbert:

<http://www2.sas.com/proceedings/sugi26/p165-26.pdf>

Introduction to GMAP by Michael Davis:

<http://www2.sas.com/proceedings/sugi22/ADVTUTOR/PAPER49.PDF>

Introduction to annotation by Keith Brown:

<http://www2.sas.com/proceedings/sugi22/INFOVIS/PAPER168.PDF>

Basics of GMAP by Mike Zdeb:

<http://www2.sas.com/proceedings/sugi29/251-29.pdf>

Advanced mapping with animation by Mike Zdeb and Bob Allison:

<http://www2.sas.com/proceedings/sugi30/137-30.pdf>

Also see Mike Zdeb's book, *Maps Made Easy Using SAS*, SAS Publishing, 2002.

Other Resources

ZCTA documentation:

http://www.census.gov/geo/ZCTA/zcta_tech_doc.pdf

<http://www.census.gov/geo/ZCTA/zcta.html>

Census Cartographic Boundary Files (shapefiles):

http://www.census.gov/geo/www/cob/bdy_files.html

Washington State Department of Transportation shapefile catalog:

<http://www.wsdot.wa.gov/mapsdata/geodatacatalog/default.htm>

Trademarks

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.

Contact Information

To receive revisions/corrections to this paper, make comments, or ask questions, please feel free to contact the author:

Deron Ferguson, Ph.D.

WA State Office of Financial Management, Olympia, WA

deron.ferguson@ofm.wa.gov

(360) 725-3862