PROC GMAP: How I Learned to Tolerate (And Almost Love) Annotating

Keith J. Brown
University of North Carolina - General Administration
Chapel Hill, NC

INTRODUCTION

Once upon a time, PROC GMAP was my most feared graphics procedure. Long after I had come to use and appreciate the GPLOT and GCHART procedures, I still shrank from attempting to use GMAP because of one word: ***annotation***. Even after I became comfortable producing basic distribution maps, anything beyond that seemed an insurmountable task because of the need to calculate the position of innumerable points. About a year ago, I was called upon to produce a map of North Carolina population distribution by county using another package, and discovered that there was no way to pass data into it. The patterns had to be applied to each of one hundred counties by clicking inside the county's border and selecting the appropriate pattern. That experience convinced me that it was time to take another look at annotating maps with SAS®. Like many of the things we fear when we are young and foolish, it turned out to be a much less daunting task than I had always thought it would be.

FINDING THE DATA

My first concern was how to get the data with which to annotate the maps. The SAS System includes several data sets that can be used for annotation, such as 'MAPS.USCENTER' (visual midpoints of each state), 'MAPS.USCITY' (major cities in each state), and 'MAPS.CNTYNAME' (county names).

While these data sets are helpful, they don't cover each state in depth. The rise of the Internet has made accessing geographical data much simpler. The "How Far Is It" site, **www.indo.com**, was the first place that I found help in finding the location of many of the cities and towns not covered by SAS' data sets. Later, I discovered that the United States Census Bureau has an electronic gazetteer of counties, incorporated places, and zipcodes available at the URL **http://tiger.census.gov/places.html** or for anonymous FTP from **www.census.gov/pub/tiger/tms/gazetteer.** These files are available in zipped, gzipped, and text format. A codebook is available at the web address. In addition to FIPS codes and names, the county and places files contain 1990 population, housing units, land and water area, and latitude and longitude.

In addition, the U.S. Geological Survey, through their Geographic Name Information Service (GNIS) has a file of over 160,000 places in the United States. This file, "**populated-places.gz"** is available in electronic form from the USGS server at **http://www-nmd.usgs.gov/pub/gnis.** While the USGS labels this file "populated places", it contains many places that are nothing more than crossroads or country stores; their compilation is truly extensive and impressive.

FINDING AN EXAMPLE

Example GR33N08 in the chapter on PROC GPROJECT answered a number of questions about annotating, such as, "What variables do I need in the annotate data set?", "How do I convert between degrees and radians?", and "How do I get the map and the annotation on the same scale?". To see the answers to those questions, let's begin with a data set of North Carolina public universities that includes the following variables:

STATE -- State FIPS Code

COUNTY -- County FIPS Code

CITY -- Name of city

NAME -- Name of university

LONG -- Longitude (in degrees)

LAT -- Latitude (in degrees)

To produce an annotation data set that would show the location of each campus by name and an asterisk, we could the following code:

```
data work.annodata;
length function style color $ 8
       position $ 1  text $ 20;
set nc.campuses(drop=state);
    retain xys ysys '2'  hsys '1';

* These set the coordinate systems *;
    retain state 100;
    state+1;
    x=long*arcos(-1)/180;
    y=lat*arcos(-1)/180;
* Convert degrees to radians       *;

    when='B';
* Annotate After or Before the map *;
* is drawn.                        *;

    function='LABEL';
    color='BLACK';
    text='T';
    style='marker';
* Prints an asterisk               *;

    angle=0; rotate=0; size=1;
* Controls the size and orientation*;
* of the text.                     *;

    position='5';
* Centers the marker               *;
output;

    state+1;
    when='A';
    color='BLUE';
    text=NAME;
    style='swissb';
* Prints the name of the school    *;

    angle=0; rotate=0; size=2;
* Controls the size and orientation*;
* of the text.                     *;

    position='E';
* Centers the name underneath the  *;
* marker.                          *;

output;
run;
```

WORK.ANNODATA should contain two observations for each university, with the variables needed to place a black asterisk at the campus' location, and to write the name of the school in blue below the marker. There is a dummy value for 'STATE' to insure that all the annotate observations are handled properly by PROC GPROJECT. Each name will be written horizontally (angle=0), and each letter in the name will not be rotated (rotate=0); the font size for the name will be twice that of the marker (size=2 vs size=1). If we needed to write the name vertically (to fit the name of a long, narrow county inside its boundaries, for example), we would need to set the angle to 270 and the rotation to 90.

If you will be using the same data set frequently for annotating maps, you should consider creating permanent variables for angle, rotation, and position. Very often, you will find that there is an optimal orientation for a county, city, or other name. Having those variables already calculated whenever the data is used will reduce the time you need to spend fine-tuning your maps.

USING THE ANNOTATE MACROS

While you can create annotate data sets by assignment statements and output statements, doing so becomes awkward as the number of different annotations increases. SAS provides a number of macros the simplify the creation of annotation data sets; before these can be used, however, they must first be compiled; include the statement '**%annomac;**' at the beginning of your code.

Among the most useful of the annotation macros are:

%LABEL -- *prints text or symbols at or around a given point*
%CIRCLE -- *draws a circle with a given radius around a given point*
%SLICE -- *draws a circle or arc with a given radius around a given point*
%LINE -- *draws a line between two given points*

LABELLING POINTS

For instance, using the %LABEL macro, the sample program above could be reduced to:

```
data work.annodata;
length function style color $ 8
       position $ 1  text $ 20;
set nc.campuses(drop=state);
    retain xys ysys '2'  hsys '1';
    retain state 100;

    x=long*arcos(-1)/180;
    y=lat*arcos(-1)/180;

    state+1;
    when='B';
%label(x,y,'T',black,0,0,1,marker,5);

    state+1;
    when='A';
%label(x,y,name,blue,0,0,2,swissb,E);
run;
```

The parameters needed for the %LABEL macro are, in order:

x -- *longitude in radians;*
y -- *latitude in radians;*
text -- *the text to be printed;*
color -- *color of the text;*
angle -- *orientation of the label;*
rotate -- or*ientation of each letter;*
size -- *font size;*
style -- *font type;* and
position -- *position of the text around the point.*

DRAWING CIRCLES AND ARCS

The General Assembly has decreed that each campus in the University of North Carolina system shall have a service area extending thirty miles from its campus. Maps depicting these service areas are in high demand at General Administration; the '%circle' and '%slice' macros make producing such maps relatively simple. If we took the preceding code and added a few lines

```
   state+1;
    when='A';
%circle(x,y,15,red);
```

before the 'run' statement, we would fulfill the Legislature's mandate by drawing a red circle 30 miles in radius around each of the sixteen campuses. The only parameters needed are the x and y locations and the size and color of the circle.

If we wished to fill in the circle, either with a solid color or a pattern, we would replace the '%circle' macro with the '%slice' macro:

```
%slice(x,y,0,360,15,red,p3n45,none);
```

This line would draw the same red circle, but would fill it in with medium thick diagonal red lines. The parameters needed are **x**, **y**, and **angle**, as before; **rotate**, which now determines how much of an arc to draw (360 degrees, in this case); **size** and **color**, also as before; **style**, which is now the pattern to be used for coloring in the circle; and **line**. This last variable controls what sort of line, if any, is drawn from the radius to the edge of the circle for arcs.

Unfortunately, there doesn't seem to be a convenient way of telling SAS that the circle or slice should have a radius of thirty miles; the fact that a size of '15' yields a circle of 30 miles is more a function of luck and North Carolina geography than of programming. Producing a circle with a specific diameter or radius is a process of trial and error.

DRAWING LINES

In addition to labelling points and drawing circles around them, it is often useful to draw lines between pairs of points by using the '%line' macro.

```
   state+1;
    when='A';
    lat1=35.9347;
    long1=79.0672;
* Location of Chapel Hill          *;
    x1=long1*arcos(-1)/180;
    y1=lat1*arcos(-1)/180;
%line(x,y,x1,y1,green,2,0.2);
run;
```

This code, added to our previous program, would draw a green line from each campus' location (x,y) to Chapel Hill (x1,y1). The line would be dotted (line pattern 2) and fairly thin (size=0.2). This macro is also useful when you have a long name for a small area (such as Rhode Island or Pasquotank County); you can use this macro to move the label to a more convenient location, and draw a line from the label back to the area being labelled.

PROJECTING AND MAPPING

Once you have built your annotation data set, the next steps are to combine it with the mapping coordinates for the boundaries; project the combined data; split the annotation data and boundary data into separate data sets; and create the map from the boundary data, annotation data, and response data. Because our focus here is annotation, we will create a dummy data set for the response data.

```
proc sort data=maps.states
    (where=(state=37 and density<=3))
    out=ncmap;
    by state;
* Select only North Carolina       *;

data all;
   set ncmap annodata;
run;
* Combine boundaries and           *;
*  annotations                     *;

proc gproject data=all  out=allproj;
   id state;
run;

data annodata ncmap;
set allproj;
if state > 100 then output annodata;
   else output ncmap;
run;

* Divides the data back into       *;
* annotating data & mapping data   *;
```

```
data dummy;
    state=37;
    count=1;
    output;
run;

pattern1 v=mempty repeat=100 c=black;

proc gmap data=dummy map=mapdata all;
     id state;
    choro count / nolegend
                  annotate=annodata;
title1 j=c c=black h=4
    "PROC GMAP";
run;
quit;
```

CONCLUSIONS

While mapping in SAS is not a "point, click, and drag" affair, that is not altogether a disadvantage. By relying on separate data sets for mapping coordinates, data to be displayed, and annotation to enhance the output, SAS makes it possible to mass produce maps in a way that other packages do not. With the increased availability of geographic data and the use of SAS macros, customizing maps to tell the right story is no longer a chore to be feared.

REFERENCES

In addition to the Internet sites mentioned above, the following are invaluable resources for annotating maps in SAS

SAS/GRAPH® Software: Reference, Version 6, First Edition, Volume 1, Cary, NC: SAS Institute, Inc., 1990, pp 465-596.

SAS/GRAPH® Software: Reference, Version 6, First Edition, Volume 2, Cary, NC: SAS Institute, Inc., 1990, pp 1001-1062 & 1147-1168.

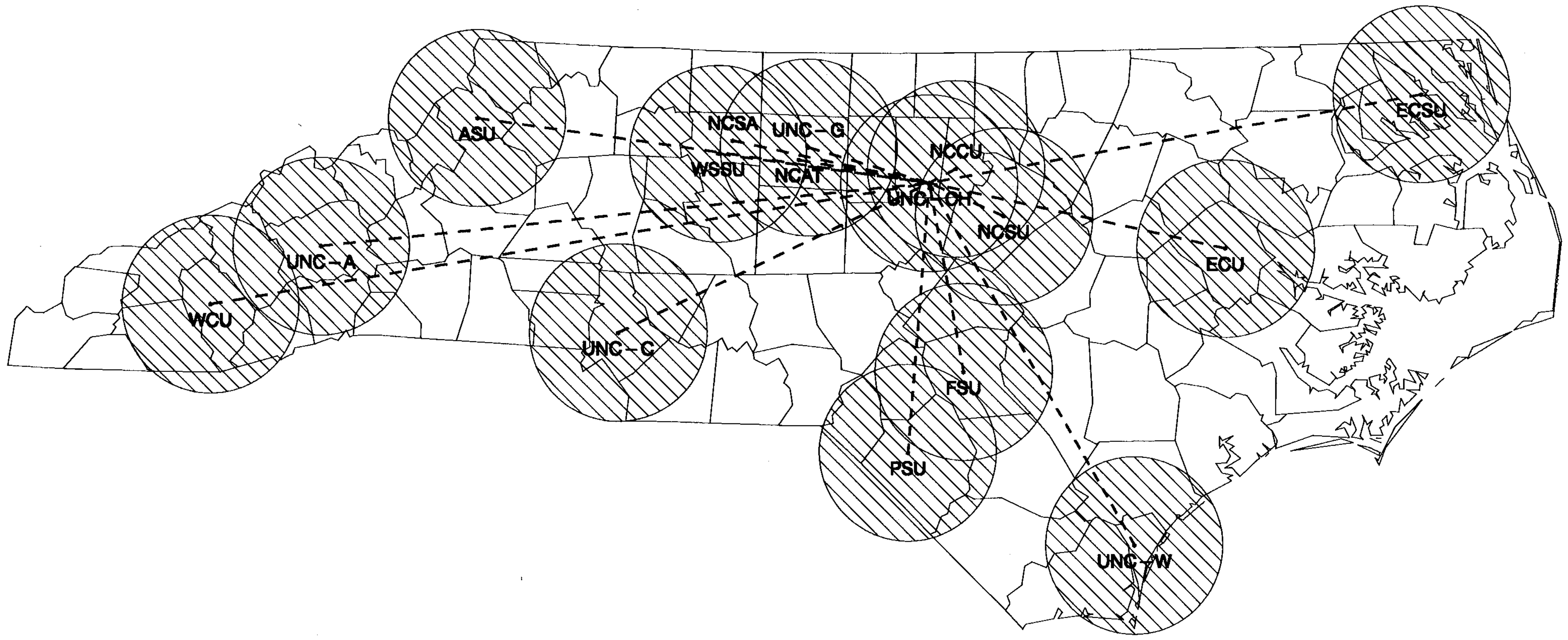
For Further Contact

The author can be reached for comments or questions at:

Keith J. Brown
P.O. Box 2688
UNC-General Administration
Chapel Hill, NC 27515-2688

(919) 962-1000 (Telephone)
(919) 962-4316 (Fax)
Keith_Brown@unc.edu (E-mail)

PROC GMAP
How I Learned to Tolerate
(And Almost Love) Annotating



Keith J. Brown/SUGI 22/17Mar1996