

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

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

Abstract

While the graphics procedures available in the SAS® System provide an easy method of turning data into informative and eye-catching charts, graphs, and maps, fine-tuning that output by adding additional information is not as simple as pointing, clicking, and drawing. This presentation will focus on using macros and electronically-available data to create annotated datasets that modify the output from the GMAP procedure by connecting points, drawing circles, and adding labels.

Introduction

Once upon a time, PROC GMAP was my most feared graphics procedure. Long after I had come to use and appreciate GPLOT and GCHART, 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 in SAS. Like many of the things we fear when we are young and foolish, it turned out to be much less daunting a 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. SAS includes several datasets 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 datasets are helpful as starting points, 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' datasets. Later, I discovered that the U.S. Census Bureau has a gazetteer of counties, places, and zipcodes in the United States available on the World-Wide Web at 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 from the USGS server at

www.nmd.usgs.gov/pub/gnis; it will need to be gunzipped before using. The README file provides a description of the file layout. 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, although some places have locations that put them outside their county boundaries as drawn by SAS.

Finding an example

The next step in overcoming my fear of annotation was the discovery of an example in the manual of exactly what I wanted to do -- GR33N08 in the chapter on PROC GPROJECT. This sample program answered a number of questions about annotating, such as, "What variables do I need in the annotate dataset?", "How do I convert between degrees and radians?", and "How do I get the map and the annotation on the same scale?". The short answers are, respectively, "xyxs, ysys, hsys, function, when, x, y, text, angle, rotation, size, style, and position"; "radians=degrees*arcos(-1)/180"; and "use PROC GPROJECT".

To flesh out those answers a bit, let's begin with a dataset 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 dataset that would show the location of each campus by name and an asterisk, we could use the following code:

Example 1

```
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 *;
* to be used *;
  retain state 100;

  x=long*arcos(-1)/180;
  y=lat*arcos(-1)/180;
* Convert to radians *;
```

```
      state+1;
* Will be used to identify each *;
* unit *;
  when='B';
* Put the annotation 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 use set the angle to 270 and the rotate to 90.

If you will be using the same dataset 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 datasets by assignment statements and output statements, doing so becomes awkward as the number of different annotations increases. SAS provides a number of macros to simplify the creation of annotation datasets; before these can be used, however, they must first be compiled. To do so, simply include the statement '%annomac;' in your code, preferably near the beginning.

Among the most useful of these 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

%MOVE -- moves without drawing

%DRAW -- draws a line from the previous point to a given point.

The last two macros are commonly used together; the others may all be used independently.

Labelling points

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

Example 2

```
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*acos(-1)/180;
  y=lat*acos(-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 -- orientation 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 a relatively simple task. If we took the preceding code and added a few lines

Example 3

```
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. Since a complete circle would have no such line, this value applies only when drawing 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. Because North Carolina has an east-to-west distance of approximately 400 miles, a circle with a diameter of 15% of that distance has a diameter of 60 miles and a radius of 30 miles. In order to produce the same circle for another state, you would need to know the state's dimensions, or find two points in that state the requisite distance apart, and experiment with various sizes until the midpoint of one circle lies on the edge of the other.

Drawing lines

In addition to labelling points and drawing circles around them, it is often necessary to draw lines between pairs of points on a map. For instance, we could draw lines from each of the campuses to Chapel Hill in a variety of ways.

Example 4

```
state+1;
when='A';
lat1=35.9347;
long1=79.0672;
* Location of Chapel Hill *;
x1=long1*acos(-1)/180;
y1=lat1*acos(-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).

Alternatively, we could use the '%move' and '%draw' macros to draw the same lines.

Example 5

```
when='A';
state+1;
lat1=35.9347;
long1=79.0672;
* Location of Chapel Hill *;

x1=long1*acos(-1)/180;
y1=lat1*acos(-1)/180;
%move(x1,y1);
when='A';
state+1;
%draw(x,y,green,2,0.2);
run;
```

The '%move' macro sets the pointer back to Chapel Hill's location as the origin for each observation, and '%draw' then draws a line to the appropriate campus.

While the results from the two methods are identical, there are times when one may be preferable to the other. If the point of origin

varies from observation to observation, you will need to use lag functions with the '%line' macro to pass the previous observation's stopping point as the current observation's starting point. With '%move' and '%draw', you would need to set the origin with '%move' at the beginning of the program, or whenever a new path, unconnected with the previous one, was beginning. If you have a long label for a small area (such as Rhode Island or Pasquotank County), you can use these macros to move the label to a more convenient location, and draw a line from the label back to the area being labelled. Another use that I've found for them is tracing the paths taken by various branches of my family as they converged on a single county in Alabama.

Post-macro processing

Since each annotate macro contains an implied 'output' statement, it is sometimes necessary to do some clean-up in a subsequent data step. While the angle, rotation, size, and color of the text can be set in the macro by using variables, position must be hard-coded. When there are several annotated points in a small area, you will often need to change the position of one or more points to ensure the map's legibility. Example 6 shows how the previous code might be changed.

Example 6

```
data annodata;
set annodata;
if name=text and function='LABEL' and
    name in('NCSA','NCCU','UNC-G')
    then position='2';
else if name=text then position='E';
run;
```

Three schools, North Carolina School of the Arts, NC Central University, and UNC-Greensboro, will have their names centered above the marker; all other schools will have their names centered below the marker. Again, if you will be working with the same annotation data frequently, you should consider creating a permanent variable for the optimal position. This would allow you to use code such as

```
if name=text then position=pos;
```

instead of changing the points individually.

Projecting the data

Once you've tailored your annotation data to your needs, you need to combine it with the dataset describing the boundaries to be mapped. If the data you have used for annotating is in the form of unprojected latitude and longitude, you should use one of SAS' unprojected boundary datasets, and project the combined data to ensure a good match. If your data has already been projected, then you should use one of the projected boundary datasets. SAS Technical Report P-196 identifies each of the boundary datasets as projected or unprojected.

PROC GPROJECT converts spherical coordinates, such as latitude and longitude, into two-dimensional coordinates in order to display geographical relationships accurately on a map. This procedure allows you to choose among several different projection methods; to clip the area of the map, regardless of the political boundaries; and to 'tilt' the angle at which you view the map. The ID statement specifies which variables define unit areas, and is required for GPROJECT. Example 7 illustrates the code needed to produce a 'flat-on' map of North Carolina. Refinements such as tilting the viewing angle will be left as an exercise to the reader.

Example 7

```
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 ncmmap;
set allproj;
if state > 100 then output annodata;
else output ncmmap;
run;

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

The map

We've now reached the point where many of my earliest experiences with PROC GMAP began; all we need is some data to be displayed on our map. Because we have so many different types of annotation to display, we will instead create a dummy dataset for our map, so that the patterns for the data don't obscure the annotations.

Example 8

```
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";
title2 j=c c=black h=3
  "How I Learned to Tolerate";
title3 j=c c=black h=3
  "(And Almost Love) Annotating";
footnote1 j=1 c=red h=2
  "_____";
footnote2 j=1 c=red h=2
  " Keith J. Brown";
footnote3 j=1 c=red h=2
  " UNC-General Administration";

run;
quit;
```

In conclusion

While mapping in SAS is not a "point, click, and drag" affair, that is not altogether a disadvantage. By relying on separate datasets 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.

SAS® Technical Report P-196, *SAS/GRAPH® Software: Map Data Sets, Release 6.06*, Cary, NC: SAS Institute, Inc., 1990.

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

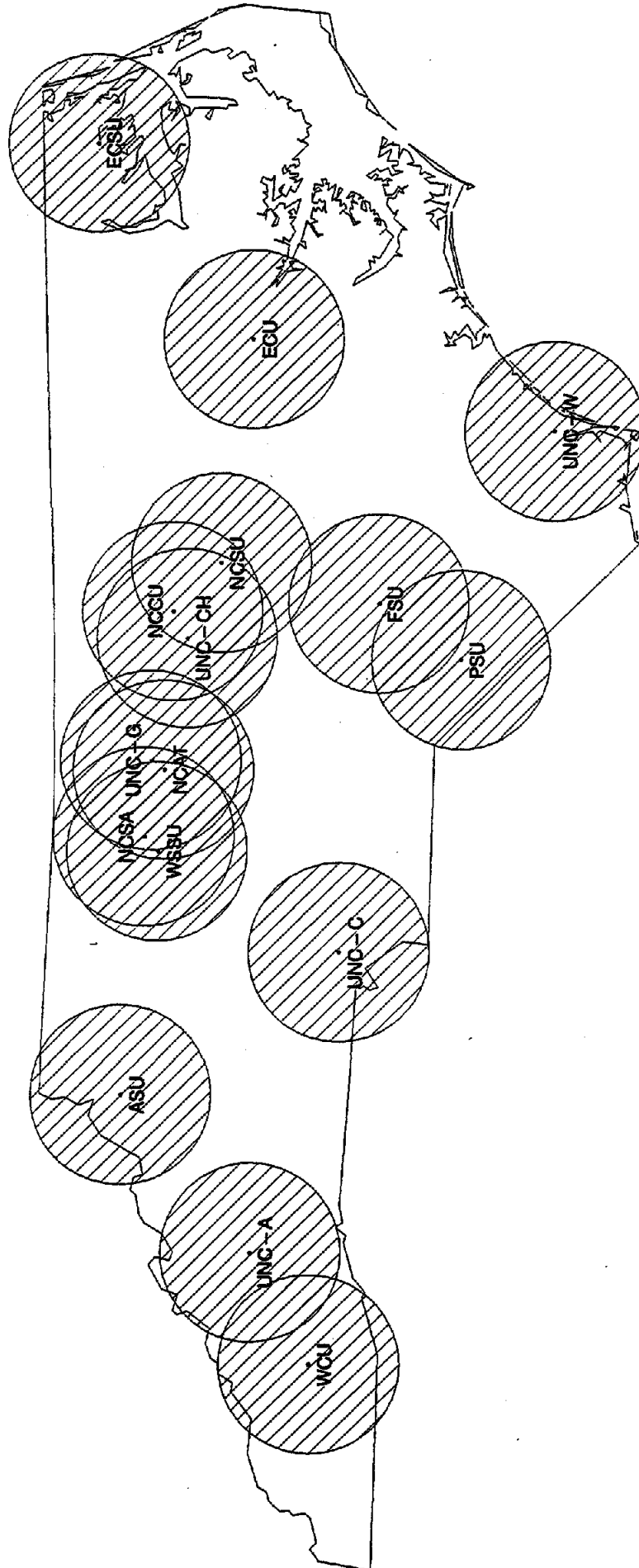
(919) 962-1000 (Telephone)
(919) 962-4316 (Fax)
Keith_Brown@unc.edu (E-mail)

SAS and SAS/GRAPH are registered trademarks of SAS Institute Inc. in the USA and other countries. ® Indicates USA registration.

PROC GMAP

How I Learned to Tolerate
(And Almost Love) Annotating

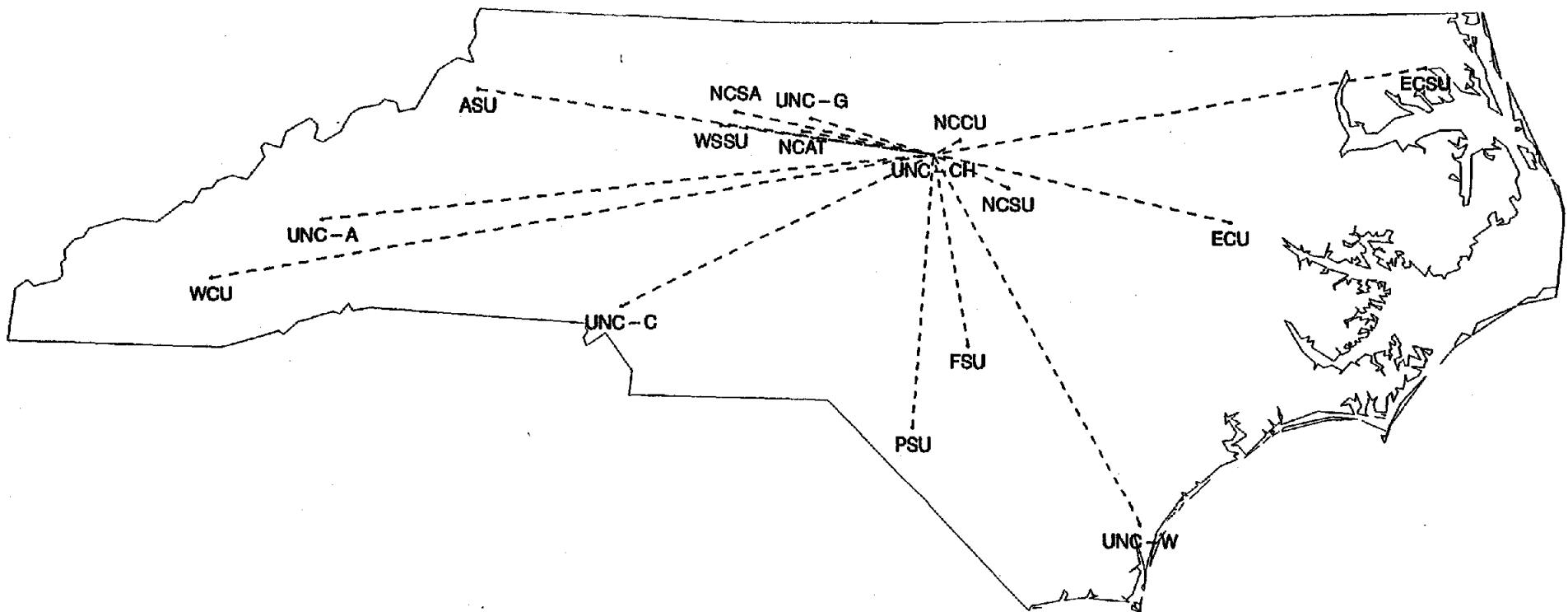
Examples 1, 2, & 3



PROC GMAP

How I Learned to Tolerate (And Almost Love) Annotating

Examples 4 & 5



481

using 1965 proceedings

user training and support

Keith J. Brown

UNC - General Administration

PROC GMAP

How I Learned to Tolerate
(And Almost Love) Annotating

Example 8

