Paper 137-30

Stretching the Bounds of SAS/GRAPH® Software

Mike Zdeb, University@Albany School of Public Health, Rensselaer, NY
Robert Allison, SAS Institute Inc., Cary, NC

ABSTRACT

This hands-on session will teach you how create graphics that you never imagined you could produce with SAS/GRAPH. You will learn some little-known, but very powerful, SAS/GRAPH tricks that you can use to create unique, custom, one-of-a-kind graphics.

Some of the areas covered during this hands-on workshop are: using PROC GMAP to create custom 'maps' that do not represent geographical areas (silicon wafer 'maps', calendar charts, floor plans, textile fabrics, company logos, data grids); using the new device=JAVAMETA graphics device to produce animated charts and slide shows (the person viewing the output can stop, start, and control the speed of the animation to help explore the data in detail, not possible with old-style gif animations); using the Annotate facility in non-standard fashions (to put 'shadows' behind maps, emphasize 2-level of borders around regions in maps, fake transparent/shadow effects in maps); using the HTML= option to add custom CHARTTIPS (sometimes referred to as 'popups') and DRILLDOWNS to maps, bars, markers, annotated polygons, annotated 'invisible' areas, and annotated images (you will also learn how to use device=JAVAMETA CHARTTIPS, generally faster than the standard HTML tips and having some extra functionality).

The workshop features numerous hands-on examples to let you learn the basic concepts, familiarize yourself with the SAS code, and see how you can easily modify the examples to use them with your own data. Finally, there are several 'super-fancy' examples that put all of the basics together in a way that are guaranteed to produce... "Wow! I did not know that you could do that with SAS/GRAPH!"

INTRODUCTION

Each of the four detailed examples in this workshop involves the use of PROC GMAP. The first and third examples actually produce maps, while the second produces a calendar, and the fourth a display of silicon wafer data. Example one uses data from the US Census Bureau web site and SAS/GRAPH animation to show the westward movement of the US population center from 1790 to 2000. The second example uses counts of deaths by day over a period of ten years to demonstrate how time trends and outliers can be detected using PROC GMAP. The third example demonstrates a number of uses of the Annotate facility in producing a map of Pennsylvania, while the last example uses a combination of PROC GMAP, Annotate, and PROC GREPLAY to draw silicon wafers.

ANIMATION

A single map is good method of showing the spatial distribution of data at one point in time. A series of maps can be used to show the change in the spatial distribution of data over time. When the series of maps is shown in succession, with each new map replacing the display of the last map, you have an animated display. Though you may not have given it much thought, you do see animated maps on a regular basis since they are commonly used to show the progression of weather fronts, hurricanes, tornados, and other features on weather forecasts.

SAS/GRAPH offers two methods for creating animated graphical output: the GIFANIM device driver; the JAVAMETA device driver. Each method has advantages. Output created with the GIFANIM device driver is easily viewed on most computers. Output created with the JAVAMETA driver requires a SAS-supplied application for viewing, but it can include on-screen controls for controlling animation speed and for viewing the animation one frame at a time using a standard web browser.

The following data step creates a SAS data set (POPCTR) using data from the US Census Bureau on the location of the US population center. The data set is to be used to add markers to a map, with each marker placed at the longitude and latitude of the US population center in any given year. Therefore, many of the variables in the data set are Annotate data set variables. If you are not familiar with Annotate facility within SAS/GRAPH, it can be used to customize SAS/GRAPH output in ways that are not possible with PROC GMAP options alone (see the REFERENCES at the end of the paper for suggested reading to learn more about the Annotate facility).

ANIMATE01.SAS
```
* create an Annotate data set - used to place a symbol at the
* location of the US population center in each of the years in the DATALINES file;
data popctr;
   retain function 'label' xsys ysys '2' hsys '3' position '5'
          size 3.5 color 'black' style 'marker' text 'C' when 'a';
   infile datalines dsd;
   input year : $4. y x location : $40.;
   fips = stfips(scan(location,-1));
   y    = y*constant('pi')/180;
   x    = x*constant('pi')/180;

   datalines;
   1790,  39.275,  76.187,  "KENT COUNTY,MD"
   1800,  39.268,  76.943,  "HOWARD COUNTY,MD"
   1810,  39.192,  77.620,  "LOUDON COUNTY,VA"
   1820,  39.095,  78.550,  "HARDY COUNTY,WV"
   1830,  38.965,  79.283,  "GRANT COUNTY,WV"
   1840,  39.033,  80.300,  "UPSHUR COUNTY,WV"
   1850,  38.983,  81.317,  "WIRT COUNTY,WV"
   1860,  39.008,  82.813,  "PIKE COUNTY,OH"
   1870,  39.200,  83.595,  "HIGHLAND COUNTY,OH"
   1880,  39.069,  84.661,  "BOONE COUNTY,KY"
   1890,  39.199,  85.548,  "DECATUR COUNTY,IN"
   1900,  39.160,  85.815,  "BARTHOLOMEW COUNTY,IN"
   1910,  39.170,  86.539,  "MONROE COUNTY,IN"
   1920,  39.173,  86.721,  "OWEN COUNTY,IN"
   1930,  39.064,  87.135,  "GREENE COUNTY,IN"
   1940,  38.948,  87.376,  "SULLIVAN COUNTY,IN"
   1950,  38.804,  88.369,  "CLAY COUNTY,IL"
   1960,  38.600,  89.210,  "CLINTON COUNTY,IL"
   1970,  38.463,  89.706,  "ST CLAIR COUNTY,IL"
   1980,  38.137,  90.574,  "JEFFERSON COUNTY,MO"
   1990,  37.872,  91.215,  "CRAWFORD COUNTY,MO"
   2000,  37.697,  91.810,  "PHELPS COUNTY,MO"
   ;
run;
```

All the variables in the RETAIN statement are to be used by the Annotate facility to place markers (in this case a solid triangle, character 'C' in the SAS/GRAPH MARKER software font) on a map. Since the latitude and longitude (variables Y and X) are expressed as degrees, they are converted to radians to match the manner in which latitude and longitude are present in the SAS map data set STATES that is used to draw the map. Also, variable FIPS is created using the SCAN and STFIPS functions (for example, the SCAN function returns the value MD in the first observation, then the STFIPS function returns a value of 24, the F(ederal)I(nformation)P(rocessing)S(tandards) code for Maryland. That code matches the state code used in the SAS map data set STATES.

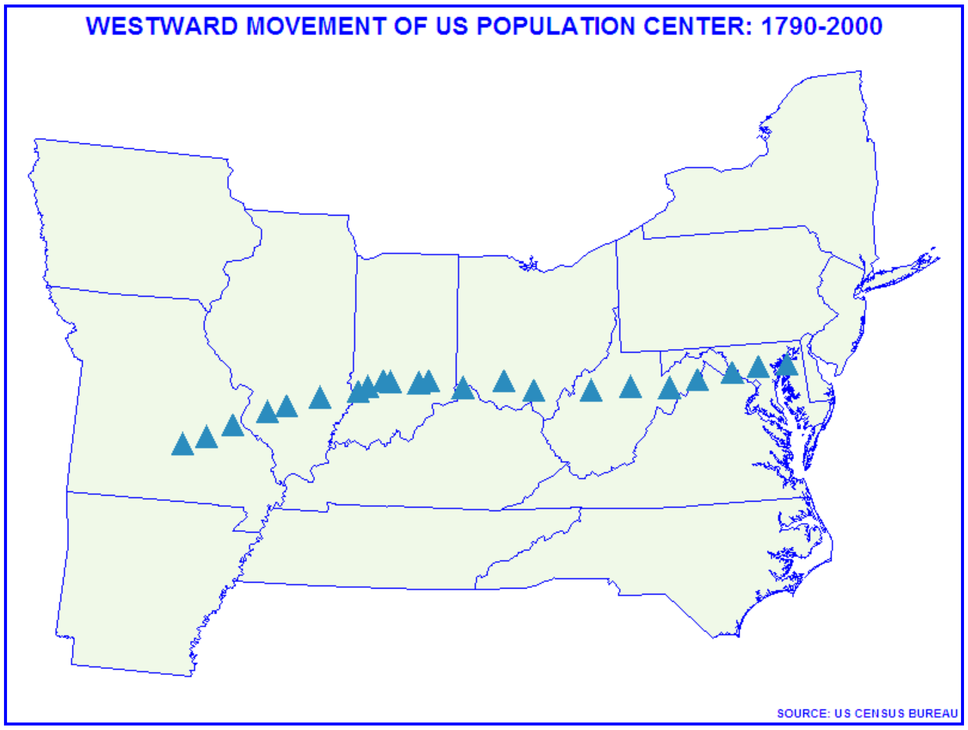
The next step is to combine the POPCTR data set with a portion of the SAS map data set STATES. In addition to limiting the number of states, the variable DENSITY is used to limit the number of observations to only those that provide enough detail needed to draw the state boundaries in a recognizable form. The combined data set is projected (PROC GPROJECT) then separated into a map data set and an Annotate data set (see the REFERENCES at the end of the paper for suggested reading to learn more about the map projection).

ANIMATE02.SAS

```
* combine the Annotate data set with selected state boundaries from
* the STATES map data set;
data both;
   set
   maps.states (where=(fipstate(state) in
                       ("IA" "IL" "IN" "OH" "PA" "MO" "KY" "WV" "VA"
                        "TN" "NC" "MD" "DE" "DC" "AR" "NJ" "NY")
                        and density le 3)) popctr;
run;

* project the combined data set;
proc gproject data=both out=bothproj;
id state;
run;

* separate the projected Annotate data set from the projected map data set;
data pmap pop;
   set bothproj;
   if when eq 'a' then output pop;
   else                output pmap;
run;
```

The projected map data set and Annotate data set can now be used to create a map. Prior to creating the map with PROC GMAP, several graphics options are selected with a GOPTIONS statement. There are seventeen states (really sixteen states plus the District of Columbia) to be drawn, so the repeat option is used in the PATTERN statement. The LS options is used on both the TITLE and FOOTNOTE statements to move the text away from the upper and lower borders.

ANIMATE03.SAS
```
goptions reset=all gunit=pct ftext="Arial/bo" ctext=blue;
* the pattern is repeated 17 times;
pattern value=ms color=cxf0f9e8 r=17;

title h=4 c=blue "WESTWARD MOVEMENT OF US POPULATION CENTER: 1790-2000" ls=.5;
footnote h=2 j=r "SOURCE: US CENSUS BUREAU " ls=.5;

* create the map with markers added with an Annotate data set;
proc gmap map=pmap data=pmap all;
id state;
choro state / discrete nolegend outline=blue annotate=pop;
run;
quit;
```

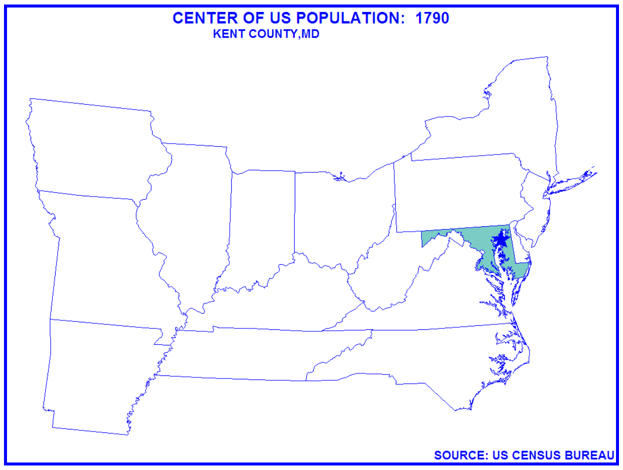


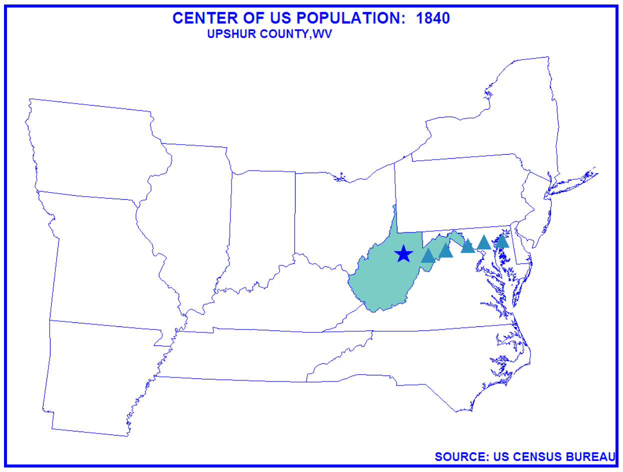
The next step is to replace the above map with a series of twenty-two maps, one for each ten year period from 1790 to 2000. Since we will be repeating the same step twenty-two times, we will use a macro to automate the process of creating all the maps. The GIFANIM device driver is used and all maps are placed in the same output file. The output file is opened in REPLACE mode and ITERATION and DELAY options are set to show the animation one time (ITERATION=1) with a 1 second delay between maps (DELAY =100, in units of .01 seconds). After the first map is written to the file, the mode is changed to APPEND. After the last map is placed in the output file, the GEPILOG option is used to add a hex character to the end of the file (a file terminator that allows the animation to display properly).

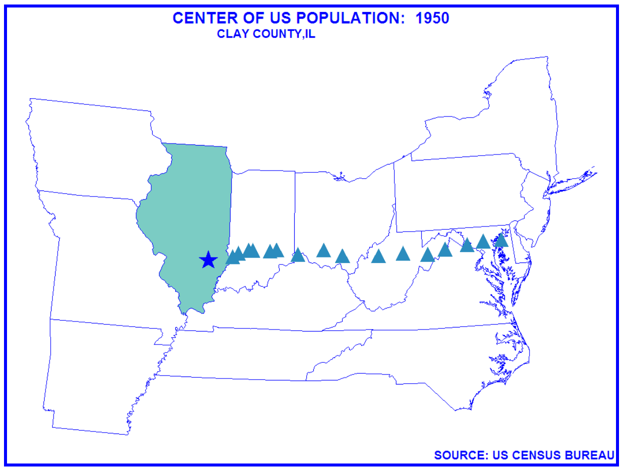
ANIMATE04.SAS
```
goptions device=gifanim gsfname=animout gsfmode=replace gunit=pct ftext="Arial/bo" ctext=blue
         htext=3 htitle=4 border iteration=1 delay=100;

* all output is written to one file;
filename animout 'z:\popctr.gif';
pattern v=ms c=cx7bccc4;

* a macro is created to repeat PROC GMAP twenty-two times;
%macro manymaps;
%do i=1 %to 22;
%if &i eq  2 %then goptions gsfmode=append;;
%if &i eq 22 %then goptions gepilog='3B'x;;
```

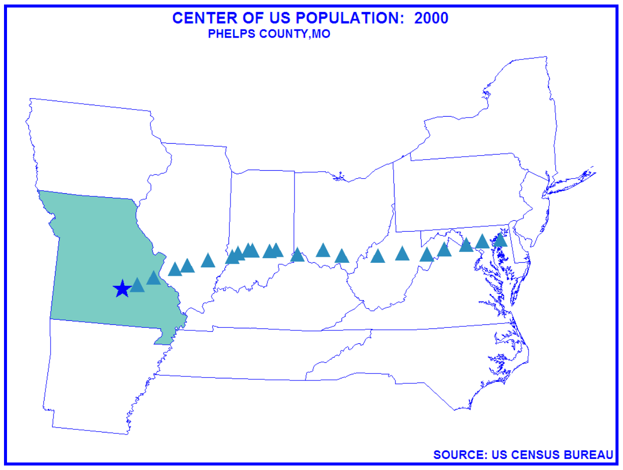
```
data anno;
length text $40;
set pop (obs=&i) end=last;
if last then do;
    size  = 5;
    color = 'blue';
    style = 'marker';
    text  = 'V';
    call symput('year' ,year);             /* macro variables YEAR and COST are used in the titles */
    call symput('cost' ,location);
    call symput('fips' ,put(fips,z3.));    /* macro variable FIPS is used in PROC GMAP */
end;
run;

title1 "CENTER OF US POPULATION:  &YEAR" ls=.5;
title2 "&cost";
footnote j=r "SOURCE: US CENSUS BUREAU " ls=.5;

proc gmap map=pmap data=pmap (where=(state=&fips))all;
id state;
choro state / discrete nolegend coutline=blue annotate=anno;
run;
quit;
%end;
%mend;

%manymaps;
```

Notice that the Annotate data set changes in size for each iteration of the macro loop. The number of markers shown on each map varies by iteration and the last marker for each iteration is always replaced by a star (character 'V' in the SAS/GRAPH MARKER software font). Also, only one state at a time is present in the response data set (DATA=MAP) in PROC GMAP. That state is highlighted with a solid pattern. A few frames of the animation look as follows.









An alternative method for creating animated graphical output is the JAVAMETA device driver. The basics are the same as when using the GIFANIM device driver in that a series of twenty-two maps is written to a single file. However, the file that is created contains metagraphics codes that require a use of the SAS-supplied file METAFILE.ZIP to interpret the codes for display in a browser. In this example, the file METAFILE.ZIP is located in the folder J:\SUGI30.

ANIMATE05.SAS

```
goptions device=javameta gunit=pct ftext='HelveticaBold' ctext=blue htext=3 htitle=4 border;

filename _webout "z:\popcenter.htm";

* write records to an HTML file
* output from the GMAP procedure will be added to the end of the HTML file
* specify the location of the file METAFILE.ZIP in the <applet code... line;
data _null_;
   file _webout;
   input;
   put _infile_;
   datalines;
   <html>
   <head>
   <title>JAVA METAGRAPHICS</title>
   </head>
   <body>
   <applet code="MetaViewApplet.class" codebase="file:///j:\sugi30\applets"
           archive="metafile.zip" Width=800 Height=600>
   <param name="BackgroundColor"    value="0xffffff">
   <param name="ZoomControlEnabled" value="false">
   <param name="Metacodes"          value="
   ;
run;

pattern v=ms c=cx7bccc4;

* a macro is created to repeat PROC GMAP twenty-two times;
%macro manymaps;
%do i=1 %to 22;

data anno;
   length text $50;
   set pop (obs=&i) end=last;
   if last then do;
       size  = 5;
       color = 'blue';
       text = 'V';
       call symput('year',year);
       call symput('cost',location);
       call symput('fips',put(fips,z3.));
   end;
run;

title1 "CENTER OF US POPULATION:  &YEAR" ls=.5;
title2 "&cost";
footnote j=r "SOURCE: US CENSUS BUREAU " ls=.5;

proc gmap map=pmap data=pmap (where=(state=&fips)) all;
   id state;
   choro state / discrete nolegend coutline=blue annotate=anno;
   run;
quit;

%end;
%mend;

%manymaps;

* write more HTML code to complete the file;
data _null_;
   file _webout mod;
   input;
   put _infile_;
   datalines;
   ">
   </applet>
   </body>
   </html>
   ;
run;

filename _webout;
```

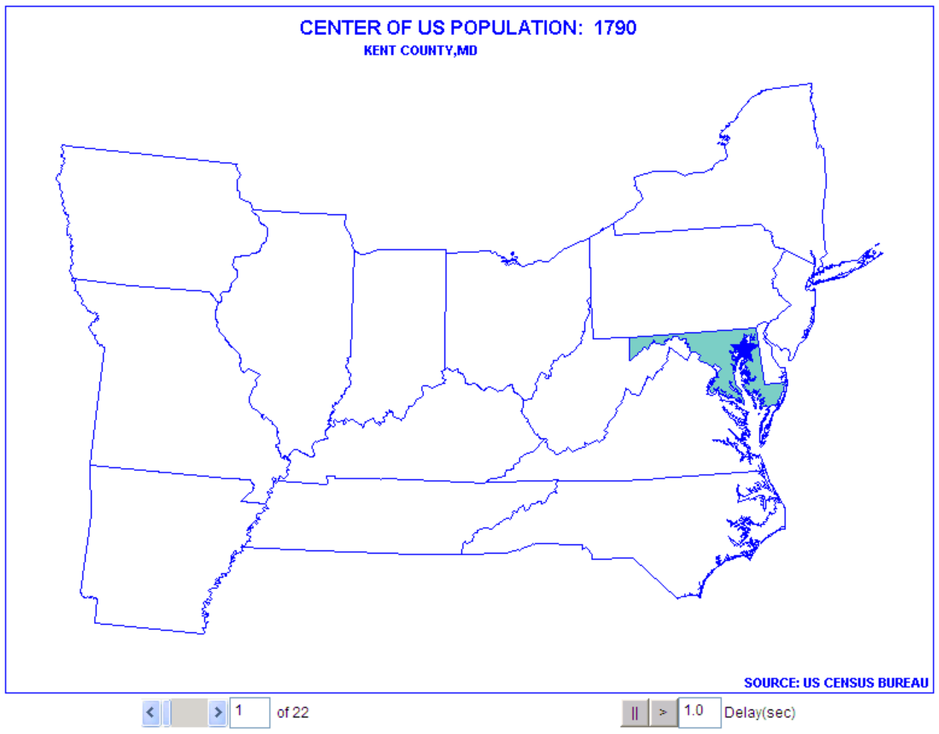
The GOPTIONS statement is similar to that used when using the GIFANIM device driver. However, a different font is used to create the annotated markers since the JAVAMETA device driver supports Postscript style fonts, not TrueType fonts. A data step writes records a file, HTML statements that will appear before the metagraphics codes written by PROC GMAP with the macro. The macro %MANYMAPS is almost identical to that used in the GIFANIM examples. The only change is removal of the two statements...

```
%if &i eq  2 %then goptions gsfmode=append;;
%if &i eq 22 %then goptions gepilog='3B'x;;
```

After the macro creates the twenty-two maps, another data step write additional statements to the end of the file, a portion of which is shown here.

```
<html>
<head>
<title>JAVA METAGRAPHICS</title>
</head>
<body>
<applet code="MetaViewApplet.class"
        codebase="file:///j:\sugi30\applets"
        archive="metafile.zip" Width=800 Height=600>
<param name="BackgroundColor"    value="0xffffff">
<param name="ZoomControlEnabled" value="false">
<param name="Metacodes"          value="
   37    8  106   97  118   97  109  101  116   97   30    0   10    1   13    5
    0    0    0   50    8   32   32   32   32   32   32   32   32   51   18   57
   46   48   49   46   48   49   77   51   80   48   55   50   56   50   48   48
   52   52    0   62    1   63    0    0   64   51   33   65 2409 1785   67    0
--- more codes ---
  45    4  523  691  578  691  551  746  523  691   61    1   60    1   61    1
   61    1   61    1   66   64   45   11  494  740  502  713  530  713  508  696
  516  671  494  685  472  671  480  696  458  713  486  713  494  740   61    1
   33   32 0000000000000000000000000000000000000000000000000000000000000000000000
">
</applet>
</body>
</html>
```

When the file is opened in a web browser, it looks as follows.



The control on the left below the map allows the display of one frame at a time, while the controls on the right are used to play and pause the animation. The delay between frames (now 1.0 second) can be changed by entering a different number in the box. If the animation is to be posted on a web site, access to the file METAFILE.ZIP can be specified as being via the web by changing a portion of the code in the first data step in the example ANIMATE05.SAS to read as follows.

```
<applet archive="http://webservername/metafile.zip"  code="MetaViewApplet.class"
                 width="800" height="600" align="TOP">
```

to specify the location of the file METAFILE.ZIP on the web server. See the REFERENCES at the end of the paper for suggested reading to learn more about the animation with either the GIFANIM or JAVAMETA device drivers.

CALGRID

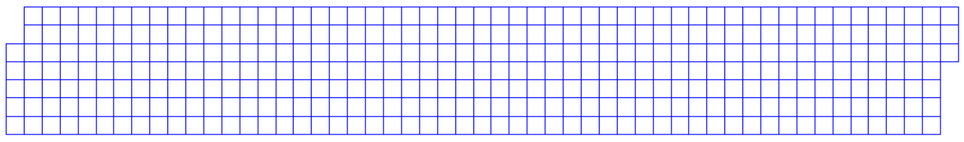
SAS/GRAPH software is supplied with many map data sets, such as countries, US states, and US counties. However, you can also create custom map data sets with your own X-Y coordinates and a variable that uniquely identifies each map area. The custom maps do not have to represent real geographical areas. They can be simple geometry that represent calendars, silicon wafers, data grids, etc. Calendar charts are a very useful way of displaying information that has a time dimension. This example shows how to: create a calendar map data set; draw the calendar with PROC GMAP; outline months and label the calendar using the Annotate facility. A custom calendar is then used to display ten years of data on the daily count of deaths that occurred in New York state.

First assume that we are creating a calendar for the year 2002. The calendar will be drawn as a grid that contains 365 rectangles (one per day). The top row of rectangles represents all the Sundays in the year, while the bottom row represents all the Saturdays. Each vertical row of rectangles represents a week. Note that the first three examples are written anticipating that the SAS code might also be used to create a multiple year calendar and prior to using the SAS code in those examples, two macro variables are created...

```
%let min_year = 2002;
%let max_year = 2002;
```

CALGRID01.SAS

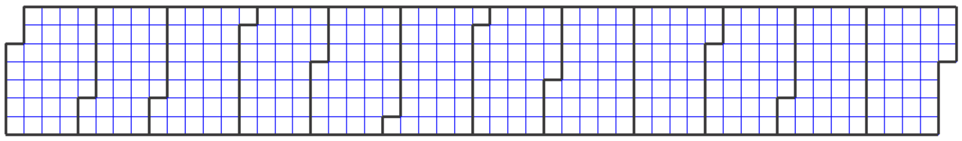
```
* create a data set with 365 observations and two variables, day and year;
data grid;
format day date7.;
do day="01jan&min_year"d to "31dec&max_year"d by 1;
   year=year(day);
   output;
end;
run;

* create a grid to be used as a map data set;
data grid;
set grid;
by year;
if first.year then x_corner=1;
else
if weekday(day) eq 1 then x_corner+1;
* if 7 is used in place of 8, there will be no space between years in a multiple year calendar;
y_corner = ((&min_year - year) * 8) - weekday(day);
x=x_corner; y=y_corner; output;
x=x+1; output;
y=y-1; output;
x=x-1; output;
keep year day x y;
run;

pattern v=e c=blue;

proc gmap data=grid (obs=1) map=grid all;
id day;
choro day / nolegend coutline=blue;
run;
quit;
```



The first rectangle in the first column (first week) and the last rectangle in the last column (last week) show that 2002 both began and ended on a Tuesday (all years except leap years start and end on the same day of the week). Each rectangle in the grid is associated with a day of the year. The next step is to outline each month using an Annotate data set. All the points in the map data set are assigned a year and month using the YYMMw. format, and PROC GREMOVE is used to create another map data set named OUTLINE that represents month boundaries. A data step is used to convert the map data set into an Annotate data set (using a slightly modified routine from SAS technical support) that can be used to add a thick border to the months in the calendar. This method is necessary since there is no automatic method for creating a two-level boundary with PROC GMAP alone (the two levels in this example being month and day).

CALGRID02.SAS

```
* add the variable yr_mon to the data set GRID;
data outline;
set grid;
yr_mon = put(day,yymmn6.);
run;

* use PROC GREMOVE to eliminate the internal boundaries (days) within each month;
proc gremove data=outline out=outline;
by yr_mon;
id day;
run;

* create an Annotate data set from the month boundaries;
data outline;
length color function $8;
retain xsys ysys '2' size 3.00 when 'A' color 'gray38' first_x first_y ;
set outline;
by yr_mon;
if first.yr_mon then do;
* save values to be used when last.year is encountered;
  first_x = x; first_y = y;
  function = 'move'; output;
end;
else do;
  function = 'draw'; output;
end;
* connect the last point to the first point;
if last.yr_mon then do;
  x = first_x; y = first_y; output;
end;
run;

pattern v=e;

* draw the grid with month outlines added using an Annotate data set;
proc gmap data=grid (obs=1) map=grid all;
id day;
choro day / nolegend coutline=blue annotate=outline;
run;
quit;
```

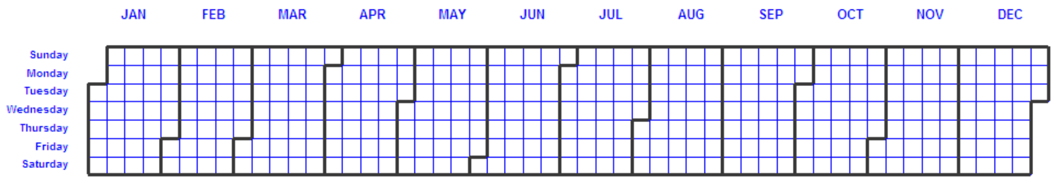


Now that the months are outlined, labels are added to the days and months by once again using an Annotate data set. A 'trick' is used to make sure that there is room for the labels (identified by a comment in the following SAS code).

CALGRID03.SAS

```
* create a data set that contains one observation for each year in the grid;
proc sql noprint;
create table years as select unique year from grid;
quit;

* create an Annotate data set containing year/month/day of week labels;
data days_months;
retain function 'label' position '4' xsys ysys '2' hsys '3'
       when 'A' style '"Arial/bo"' color 'blue' ;
length text $10;
set years end=last;
x=-6;
y=((&min_year - year)*8) - 1.25;
size = 2.0;
text=trim(left(put(year,4.))); output;
x=-.1;
size=1.5;
```

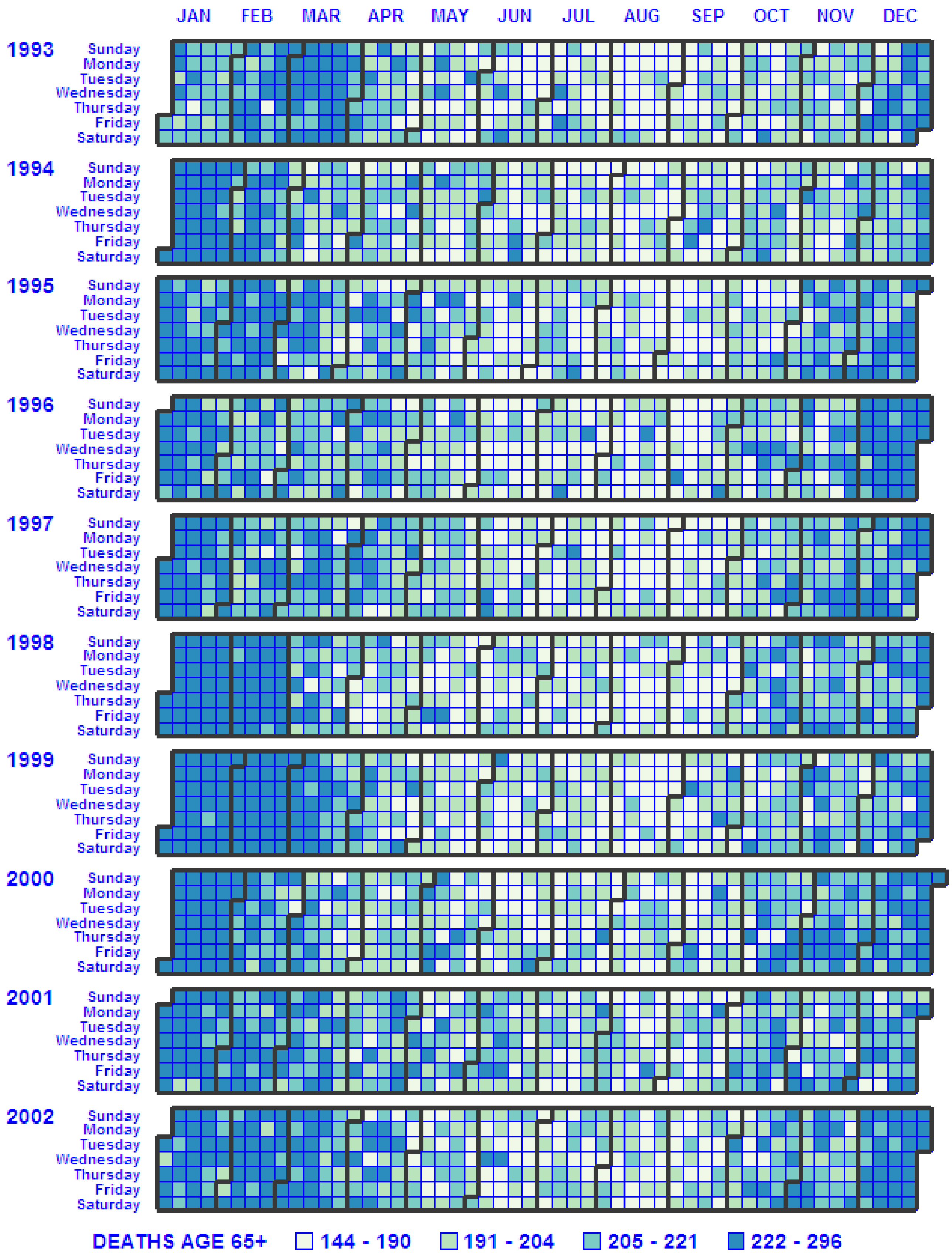
```
text = 'Sunday'; output;  y=y-1; text='Monday'; output; y=y-1; text='Tuesday'; output;
                          y=y-1; text='Wednesday'; output; y=y-1; text='Thursday'; output;
                          y=y-1; text='Friday'; output; y=y-1; text='Saturday'; output;
if last then do;
  position='5';
  size=1.75;
  y=1;
  spacing=4.4;
  x=3.5; text='JAN'; output;
  x=x+spacing; text='FEB'; output;   x=x+spacing; text='MAR'; output;
  x=x+spacing; text='APR'; output;   x=x+spacing; text='MAY'; output;
  x=x+spacing; text='JUN'; output;   x=x+spacing; text='JUL'; output;
  x=x+spacing; text='AUG'; output;   x=x+spacing; text='SEP'; output;
  x=x+spacing; text='OCT'; output;   x=x+spacing; text='NOV'; output;
  x=x+spacing; text='DEC'; output; end;
run;

* the 'trick' - add a white (unseen) grid at the top-left to create room for the Annotate labels;
data grid_plus;
set grid end=last;
output;
if last then do;
 day=1;
 color='white';
 x=-10; y=1; output;
 x=x-.001; y=y+.001; output;
 x=x+.002; output;
end;
run;

pattern v=e;

proc gmap data=grid (obs=1) map=grid_plus all anno=outline;
id day;
choro day / nolegend coutline=blue annotate=days_months;
run;
quit;
```



You can see that creating the calendar grid as a map data set, drawing it with PROC GMAP, outlining months, and labeling the calendar is very easy since all of the geometry can be easily calculated using the year/month/day of week information. Now the code will be modified slightly to create a 10-year calendar. Given that the previous three examples were all written to accommodate multiple years, the only change that is need in the SAS code is to change the values of the macro variables MIN_YEAR and MAX_YEAR. The data displayed in the calendar are the number of deaths by day among the elderly (age 65+) in New York state (except for New York City). The data set DTAB65 contains one observation for each day in the 10-year period, and two variables: day, to place the counts in the proper rectangle of the calendar; deaths, variable to be 'mapped' in the calendar.

CALGRID04.SAS

```
%let min_year=1993;
%let max_year=2002;

%include 'j:\sugi30\sas\calgrid01.sas';  /* creates the 10-year calendar */
%include 'j:\sugi30\sas\calgrid02.sas';  /* creates the month boundaries */
%include 'j:\sugi30\sas\calgrid03.sas';  /* creates the year/month/day of week labels */

pattern1 value=solid color=cxf0f9e8; /* lightest */
pattern2 value=solid color=cxbae4bc;
pattern3 value=solid color=cx7bccc4;
pattern4 value=solid color=cx2b8cbe; /* darkest */

* label the legend and change the bar shape;
legend1 label=('DEATHS AGE 65+') shape=bar(1,1.3);

* use both Annotate data sets (month outlines and year/month/day of week labels;
proc gmap data=dtab65 map=grid_plus all anno=outline;
id day;
choro deaths / levels=4 legend=legend1 coutline=blue annotate=days_months;
run;
quit;
```


JAN
FEB
MAR
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
DEATHS AGE 65+
144 - 190
191 - 204
205 - 221
222 - 296

CALGRID + CHARTTIPS

The output from CALGRID04.SAS shows a temporal pattern in the number of deaths that occur among the elderly. In addition to seeing this pattern, it would also be interesting to know both the date and the number of deaths for days that appear to be outliers (dark rectangles in the midst of light areas and vice-versa). There is not enough room to print labels on each day (rectangle), but this information can be provided using a combination of HTML output containing ALT tags and GIF output. A data step can be used to add ALT tag information to the response data set (DTAB65) used in PROC GMAP.

```
data dtab65;
set dtab65;
html_alt = 'alt="' || 'DATE: '    || put(day,mmddyy10.)            || '0d'x ||
                      'DEATHS: '  || trim(left(put(deaths,best.))) || '"';
run;
```

ODS statements and the HTML option (specifying the variable in the response data set DTAB65 that contains the ALT tag information) in PROC GMAP are added to the SAS code shown in CALGRID04.SAS to enable CHARTTIPS in the displayed output. Each occurrence of the string '0d'x (the hex character for a carriage return) produces a new line in the CHARTTIP.

CALGRID05.SAS

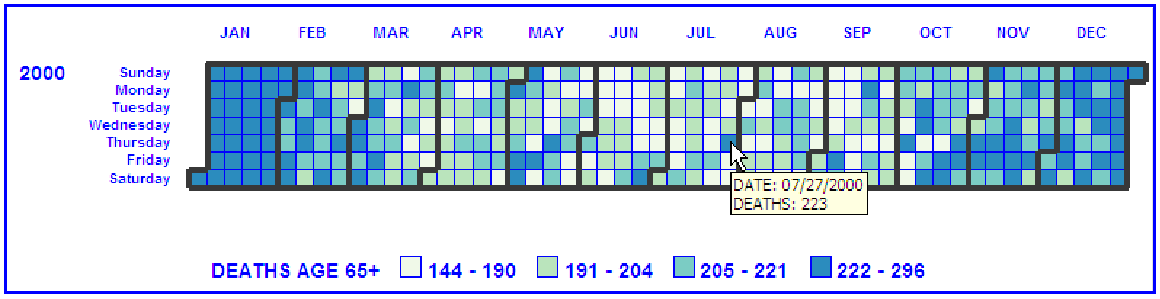
```
goptions reset=all;
%let min_year=2000;
%let max_year=2000;

%include 'k:\sugi30\sas\calgrid01.sas';  /* creates the 10-year calendar */
%include 'k:\sugi30\sas\calgrid02.sas';  /* creates the month boundaries */
%include 'k:\sugi30\sas\calgrid03.sas';  /* creates the year/month/day of week labels */

goptions ctext=blue gunit=pct htext=2 ftext="Arial/bo" device=gif xpixels=800 ypixels=800;

pattern1 value=solid color=cxf0f9e8; /* lightest */
pattern2 value=solid color=cxbae4bc;
pattern3 value=solid color=cx7bccc4;
pattern4 value=solid color=cx2b8cbe; /* darkest */

* label the legend, change the bar shape, move the legend to just beneath the calendar;
legend1 label=('DEATHS AGE 65+') shape=bar(1.75,1.75) mode=share origin=(,35) pct;

* close the listing output destination;
ods listing close;

* specify a folder for the HTML and GIF output
* URL=NONE removes any folder mention for the location of the GIF file specified in the HTML output;
ods html path='z:\' (url=none) file='calgrid.html';

proc gmap data=dtab65 map=grid_plus all anno=outline;
id day;
choro deaths / levels=4 legend=legend1 coutline=blue annotate=days_months html=html_alt;
run;
quit;

ods html close;
ods listing;
```

Notice that there is no GSFNAME option used in the GOPTIONS statement. The filename for the GIF output is provided by ODS. In the ODS HTML statement, a PATH (folder) is specified for both the HTML and GIF output files. The HTML file contains a reference to the location of the GIF file in an IMG tag. The URL=NONE option in the ODS statement ensures that no folder location is specified with the SAS-assigned name of the GIF file (gmap128.gif)...

```
<img alt="GMAP: Choropleth Map of deaths." src="gmap128.gif" border="0" usemap="#gx32b4l1_map">
```

The top 'map' on the next page shows the CHARTTIP that appears when the cursor is placed over an 'outlier' rectangle in July, 2000. The date and number of deaths are displayed. There are other device drivers that create CHARTTIPS automatically (JAVA, ACTIVEX). However, a user has little control as to the content of the information that is displayed. ODS and the GIF device driver used in concert with a data step provide a user with total control over CHARTTIP content.

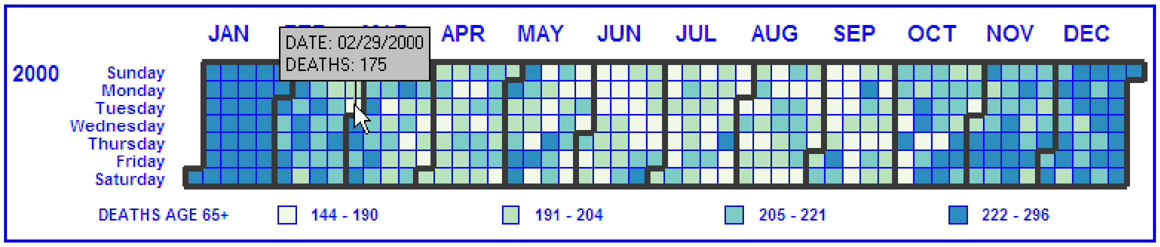
Another way to control information in a CHARTTIP is to use the JAVAMETA device driver (as was used in example ANIMATE05.SAS). Since we are only drawing one 'map' (in contrast to drawing twenty-two maps in the animation example), ODS can be used to write all the information needed to display the output. As in the previous example, a data step is used to add the CHARTTIP to the response data set.



produced with ODS plus the GIF device drive

Notice that the third %INCLUDE statement in CALGRID06.SAS specifies a different file than was used in the previous example. The new file, CALGRID03_JAVAMETA.SAS, uses a different font than CALGRID03.SAS, replacing "Arial/bo" with 'HelveticaBold'.

CALDRID06.SAS
```
goptions reset=all;
%let min_year=2000;
%let max_year=2000;

%include 'k:\sugi30\sas\calgrid01.sas';           /* creates the 10-year grid */
%include 'k:\sugi30\sas\calgrid02.sas';           /* creates the month boundaries */
%include 'k:\sugi30\sas\calgrid03_javameta.sas';  /* creates the year/month/day of week labels */

goptions ctext=blue gunit=pct htext=2.75 ftext='HelveticaBold' device=javameta xpixels=800 ypixels=800;

pattern1 value=solid color=cxf0f9e8; /* lightest */
pattern2 value=solid color=cxbae4bc;
pattern3 value=solid color=cx7bccc4;
pattern4 value=solid color=cx2b8cbe; /* darkest */

* label the legend, change the bar shape, move the legend to just beneath the calendar;
legend1 label=(j=r 'DEATHS AGE 65+') shape=bar(1.5,1.5) mode=share origin=(,40) pct;

ods listing close;

* use ODS to write the entire output file;
ods html file='z:\calgrid.html' parameters=("ZoomControlEnabled"="false" "DataTipStyle"="stick_fixed");

proc gmap data=dtab65 map=grid_plus all anno=outline;
id day;
choro deaths / levels=4 legend=legend1 coutline=blue annotate=days_months html=html_alt;
run;
quit;

ods html close;
ods listing;
```

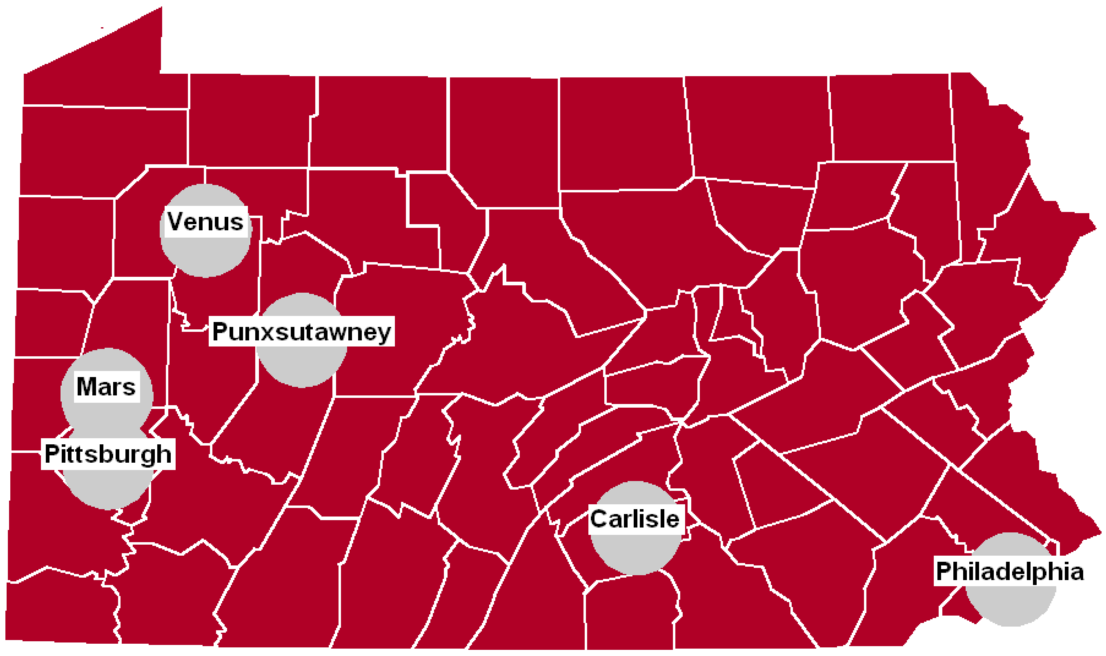


produced with ODS plus the JAVAMETA device driver

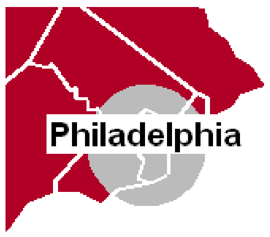
Another outlier day is chosen, in February, 2000. The date is February 29 and there were only 175 deaths. The appearance of the CHARTTIP is different from that of the ODS+GIF+ALT tag. There is a line that connects the CHARTTIP box to the center of the date rectangle and the boxes will appear on the screen much faster than in the previous example. As with JAVAMETA animation, the display depends on access to the SAS-suppled file METAFILE.ZIP. Notice that two parameters were specified in the ODS HTML statement. The first suppressed the default appearance of a zoom control at the bottom center of the output. The second results in the connection of the box to the graphics area (in this case, a date) with a line (or 'stick').

ZIPMAP

The previous examples have already introduced the Annotate facility. PROC GMAP is used again in combination with the Annotate facility to create a map of Pennsylvania with: fake transparent effects; 'shadows' behind the map; a gradient shaded backgrounds. The example starts with a fairly basic map, a county map, with Annotate markers and labels, based on longitude/latitude coordinates from SASHELP.ZIPCODE. Then, the all of the 'extras' listed above are added. Only SAS code fragments are shown in this section, but the SAS code used to produce all the maps in this section is available on request from one of the authors (see contact information at the end of the paper). The starting point for this example is the following map.



The next task is to make the city markers appear transparent. They will be put in front of the map, but made to appear 'transparent' so that you can still see the county outlines behind the markers (we do not want the markers to obscure the county outlines). To accomplish this illusion, the map is drawn and markers are added using the Annotate facility. A second copy of the county outlines is added on top of the markers by first creating an Annotate data set from the county outlines and adding the outline to the map with the Annotate facility (rather than just relying on GMAP to draw them, similar to the technique used to outline months in the CALGRID example). Various portions of the annotation can be layered, whereas without using Annotate the map and the border outlines must appear at a single depth. Here is a portion of the map showing the 'transparency'.

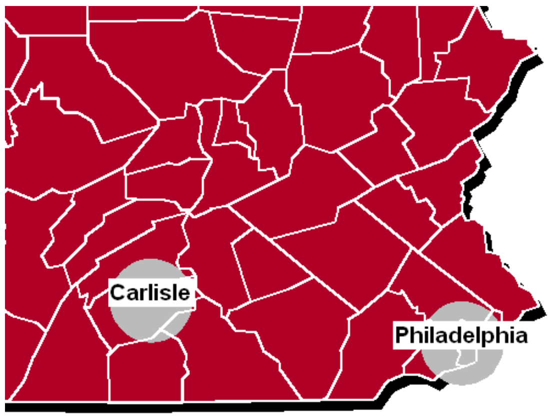


The county outlines are changed to an Annotate data set using a data step. The process is fairly simple. The data are processed by county and for each county the first observation is associated with an Annotate 'move' to the first X-Y location, while subsequent observations within a county are associated with an Annotate 'draw' to remaining X-Y points on the county border. Here is a summarized version of the code used to produce the Annotated county outlines shown in this example.

```
data border_anno;
   set my_map; by county segment;
   if first.Segment then do;
      FUNCTION = 'Move'; FX = X; FY = Y; end;
   else if FUNCTION ^= ' ' then do;
      if X = .  then do;
         X = FX; Y = FY; output; FUNCTION = ' '; end;
      else FUNCTION = 'Draw';
   end;
   if FUNCTION ^= ' ' then do;
      output;
      if last.Segment then do;
         X = FX; Y = FY; output; end;
   end;
run;
```

Projecting a shadow behind a map helps to give the illusion that the map 'leaps' off the page, helping to catch the attention of a person viewing the map. A shadow also helps to separate the borders of the map from the background. There is no option to add a shadow in PROC GMAP, but once again, the Annotate facility can be used.

PROC GREMOVE is used to create a new map with all the internal boundaries removed. The new map contains only the X-Y coordinates for the state border and it is converted into an Annotate polygon using the 'poly' and 'polycont' Annotate functions. This is done since the entire area of this polygon will be filled with a color, which cannot be done with 'move' and 'draw'. A dark color such as black usually works well for the shadow. The Annotated polygon is drawn before (behind) the map by using the ' when='b' Annotate option. A slight offset is added to the X-Y coordinates to produce the shadow shown on the right.



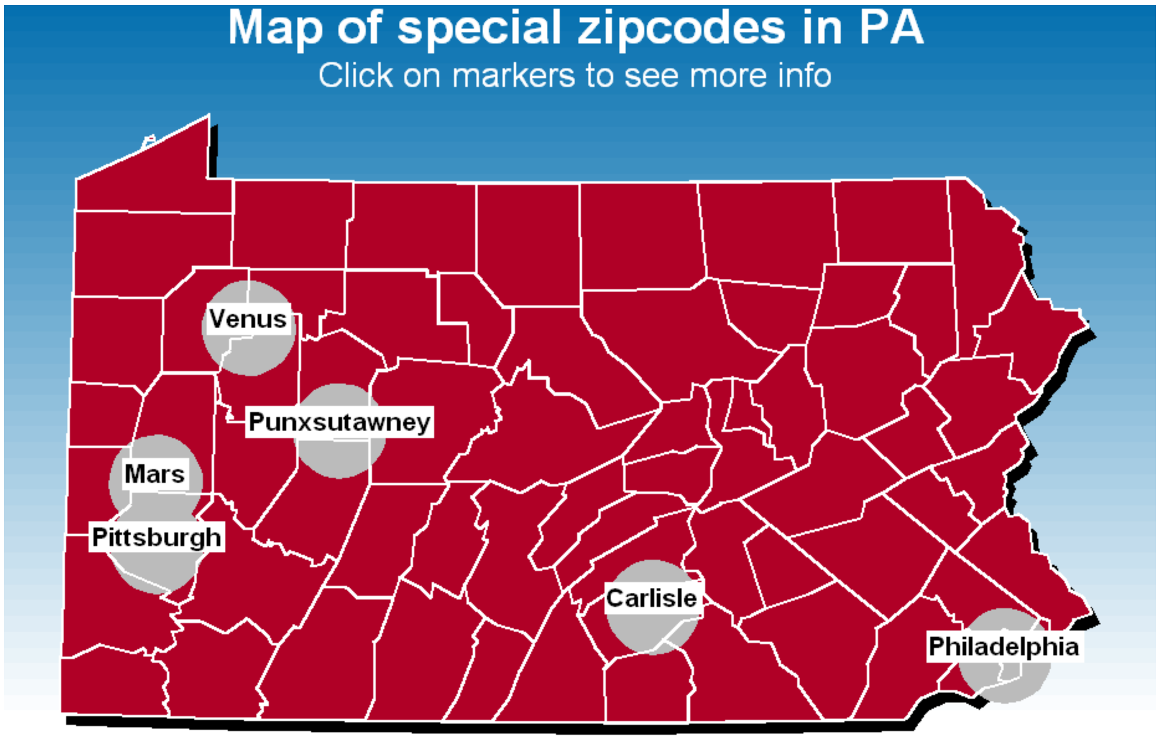
Here is a summarized version of the code.

```
* remove internal map boundaries leaving only the state
border;
proc gremove data=anno_shadow out=anno_shadow;
by county;
id county;
run;

* convert the map data set to an Annotate data set;
data anno_shadow;
set anno_shadow;
  by county segment;
   color='black';
   when='b';
   style='msolid';
   if first.segment then do;
      Function = 'poly'; fx = x; fy = y;
      end;
   else if function ^= ' ' then do;
      if x = .  then do;
         X = fx; y = fy; output; function = ' ';
      end;
      else function = 'polycont';
   end;
   if function ^= ' ' then do;
      output;
      if last.segment then do;
         X = fx; y = fy; output;
      end;
   end;
run;

* move the coordinates 'slightly' to create a shadow;
data anno_shadow;
set anno_shadow;
x=x+.0004; y=y-.0006;
run;
```

The last step is to add a gradient-shaded background. Just as there is no 'shadow=yes' option, there is no "gradient=yes" option in PROC GMAP. However, a gradient-shaded background can be produced using the Annotate facility. The gradient background is simulated by using a series of wide-and-short Annotate bars of slightly differing color. The xsys and ysys='3' coordinate system is used so the X-Y values within the Annotate data set can be specified as a percentage of the entire graphics output area, from zero to 100. The gradient starts at Y=100 and progresses to 0 in -2 increments. This technique draws fifty wide-and-short bars that start at the top of the screen and progress to the bottom of the screen.

An Annotated bar is created by an Annotate 'move' to the bottom-left corner of the bar, and then an Annotate 'bar' using the coordinates of the top-right corner of the bar. The gradient colors are specified using RGB (red/green/blue) color codes at the beginning of the loop, incrementing the colors each time through the loop. As the colors increment, they become 'lighter' and eventually fade out to white. The RGB values are kept separate for the mathematical calculations, and at the end of each pass through the loop they are converted from decimal to hex and concatenated. Putting a 'cx' in front of the concatenated hex codes provides a color name that SAS/GRAPH understands (such as 'cx0570B0'). Here is a summarized version of the code, followed by the final map.

```
data anno_gradient;
retain xsys '3' ysys '3'  when 'b' style 'solid';
orig_red   =  input('05',hex2.);
orig_green = input('70',hex2.);
orig_blue  = input('B0',hex2.);
do i=100 to 0 by -2;
 count+1;
 function='move'; x=0; y=i; output;
 function='bar'; x=100;  y=i-2;
 percent=(count-1)/(45);
 red=orig_red     + ( (256-orig_red)   * percent );
 if red > 255 then red=255;
 green=orig_green + ( (256-orig_green) * percent );
 if green > 255 then green=255;
 blue=orig_blue   + ( (256-orig_blue)  * percent );
 if blue > 255 then blue=255;
 rgb=put(red,hex2.)||put(green,hex2.)||put(blue,hex2.);
 color="cx"||rgb;
 output;
 end;
run;
```


Map of special zipcodes in PA
Click on markers to see more info
Venus
Punxsutawney
Mars
Pittsburgh
Carlisle
Philadelphia

SILICON WAFERS

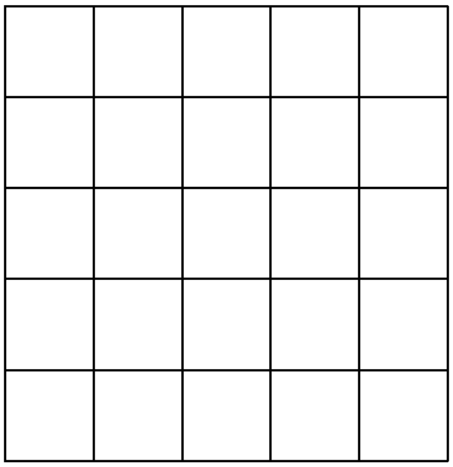
This example demonstrates how you can draw a silicon wafer map using PROC GMAP, and in addition it demonstrates how SAS/GRAPH can display several wafer maps on the same page, enabling you to see the spatial relationship of the wafer slices in the original silicon crystal from which they were cut. Only SAS code fragments are shown in this section, but the SAS code used to produce all the 'maps' in this section is available on request from one of the authors (see contact information at the end of the paper).

The data for this example assumes your silicon wafer is divided into 5-cells in the X direction and 5-cells in the Y direction, 25 cells total per each wafer (this number can be easily adjusted). Each cell has an ID and a numeric integer response value in the range 1-10. The ID value is created by concatenating the X-Y coordinates, with an underscore between them.

```
data mydata;
input plane x y value;
 length id $ 10;
 id = trim(left(put(x,best.)))|| '_' || trim(left(put(y,best.)));
datalines;
1 1 1 1
1 2 1 1
1 3 1 1
1 4 1 1
1 5 1 1
1 1 2 1
1 2 2 1
(and so on...)
;
run;
```

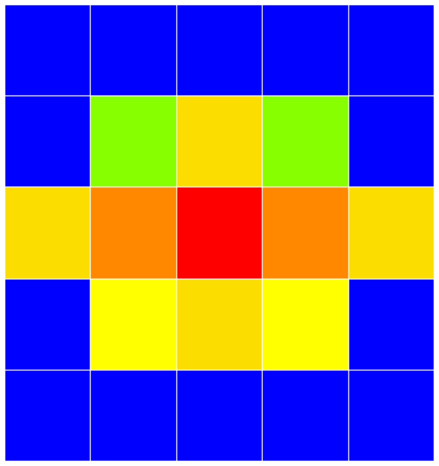
A map data set is created that can be used by PROC GMAP to draw the silicon wafer map. It must have an ID variable that corresponds to the ID in the response data, and each cell must have 4 X-Y coordinates describing the rectangular geometry of the cell (not to be confused with the X-Y coordinates used in the data above). This same map will be used to map all of the silicon wafer slices.

```
data gridmap;
length id $ 10;
 do grid_y = 1 to 5;
  do grid_x = 1 to 5;
     id=trim(left(put(grid_x,best.)))||'_'||trim(left(put(grid_y,best.)));
     x=grid_x*10-10; y=grid_y*10-10; output;
     x=x+10; output; y=y+10; output; x=x-10; output;
  end;
 end;
run;
```



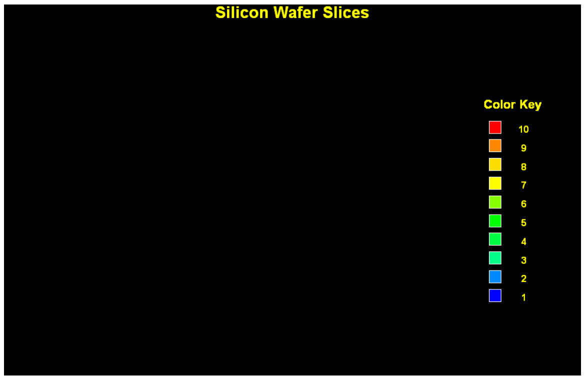
The map is shown on the right without any response/color data plotted on it. Next, the data for each of the 5 silicon wafer maps is plotted onto the map. Pattern statements are used to define the colors (ten in this example), and the 'midpoints=' are hard-coded in PROC GMAP so that each of the 10 colors gets mapped to the same response value in each of the wafer maps. The 'name=' option is used to save each of the five maps with a name that is used later to 'greplay' the maps into a custom template.

```
pattern1 v=s c=cx0000ff;
pattern2 v=s c=cx0088ff;
pattern3 v=s c=cx00ff88;
pattern4 v=s c=cx00ff44;
pattern5 v=s c=cx00ff00;
pattern6 v=s c=cx88ff00;
pattern7 v=s c=yellow;
pattern8 v=s c=cxfcdd00;
pattern9 v=s c=cxff8800;
pattern10 v=s c=cxff0000;

proc sql;
create table foo as select * from mydata where plane eq 1; quit;

proc gmap data=foo map=gridmap all;
 id id;
 choro value /
 midpoints = 1 2 3 4 5 6 7 8 9 10
 nolegend
 coutline=white
 html=htmlvar
 des='' name='plot1';
run;
```



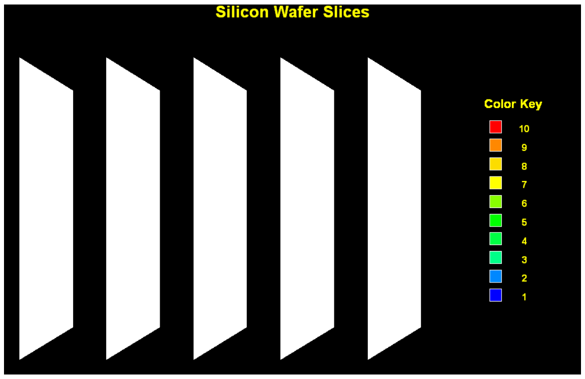
Here is an example of what one of the five wafer maps looks like, with response values mapped to the appropriate colors. No TITLE or LEGEND is used on the five silicon wafer maps. All five wafer maps are to be displayed on the same page and one TITLE and one LEGEND will be overlayed onto the display using output from PROC GSLIDE in PROC GREPLAY. The code for PROC GSLIDE is as follows (minus the code to add the legend using the Annotate facility --- a number of cleverly placed text labels and colored boxes created using the SAS/GRAPH MARKER software font 'U' character).

```
title c=yellow "Silicon Wafer Slices";
proc gslide des="" name="titles" anno=color_legend;
run;
```



The output from PROC GSLIDE is shown on the right. Five silicon wafer maps are saved by name as PLOT1 through PLOT5. A TITLE and LEGEND are saved as TITLES. A custom template is created in PROC GREPLAY with a place to draw each of the 5 wafer maps, plus a 6th section covering the entire page for the TITLE and LEGEND. The X-Y coordinates for the 4 corners of each 'hole' are defined in the GREPLAY template. Each plot is 'replayed' in the appropriate 'hole'. The code for defining the left-most of the five wafer map 'holes' (the other holes are very similar, just shifted to the right), and the code for the hole that covers the entire area (used to overlay the title/legend slide) is as follows.

```
%let backcolor=black;
proc greplay tc=tempcat nofs igout=work.gseg;
tdef stacked des='stacked'
   1/llx = 0   lly =  0
     ulx = 0   uly = 90
     urx =15   ury = 75
     lrx =15   lry = 15
     color=&backcolor

   (similarly, define the 4 other holes (holes 2-5))

   6/llx =0   lly =  0
     ulx =0   uly = 100
     urx =100   ury = 100
     lrx =100   lry = 0
     color=&backcolor
;
```

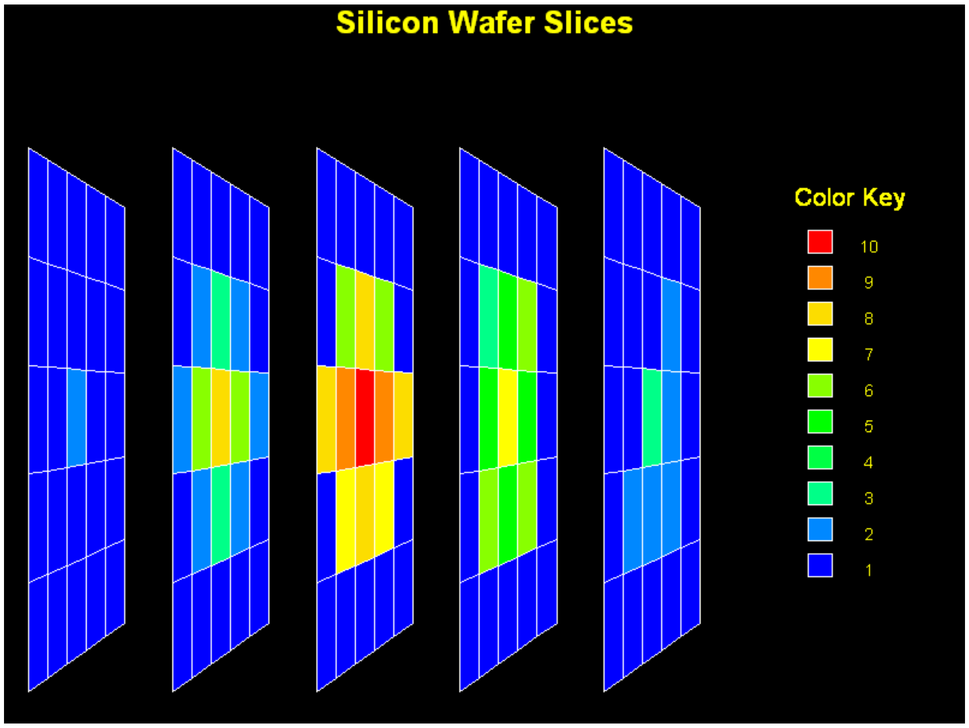


The 'trick' used for each of the five wafer map holes is that they are not rectangular. They are created so that the left side looks bigger than the right side, making them look 3-dimensional. PROC GREPLAY does all the hard work of stretching and squeezing images to fit into these non-rectangular regions ('holes'). The custom template is shown on the right.

The five wafer maps (PLOT1 through PLOT5) and the TITLE and LEGEND (TITLES) are 'replayed' into the appropriate 'holes' (1-6) in the custom template...

```
treplay 1:plot1 2:plot2 3:plot3 4:plot4 5:plot5 6:titles
        des='';
run;
quit;
```

producing the following silicon wafer map on the right.



CONCLUSION

Though SAS/GRAPH procedures can produce publish-ready output on their own, use of some extra SAS features (the Annotate facility, data step programming) and your imagination allow you to go way beyond what you normally expect as SAS/GRAPH output. PROC GMAP provides a drawing tool that can not only display conventional maps, but also non-conventional, user-defined maps such as calendars and silicon wafer slices. The ability to add animation to output allows you to easily add a time-dimension to the display of information. The easy addition of CHARTTIPS to web-based displays gives you the capability of adding information to various locations on the output. Use of the Annotate facility allows you to add extra information and special effects to graphical output.

REFERENCES

The SAS/GRAPH manual is the starting point for basic information on SAS/GRAPH procedures and on the Annotate facility. The manual is available on-line.... http://support.sas.com/documentation/onlinedoc/index.html

An additional source for information on the Annotate facility is a publication in the SAS Books-by-Users series...

Annotate: Simply the Basics
by Art Carpenter

http://www.sas.com/apps/pubscat/bookdetails.jsp?catid=1&pc=57320

PROC GMAP basics, information on other map-related PROCs such as GPROJECT, and some advanced material (including animation) can be found in another SAS Books-by-Users publication...

Maps Made Easy Using SAS
by Mike Zdeb

http://www.sas.com/apps/pubscat/bookdetails.jsp?catid=1&pc=57495

There are numerous SAS/GRAPH examples, including the SAS code used to produce them on-line at...

http://support.sas.com/techsup/sample/sample_graph.html

and more SAS/GRAPH examples at... http://support.sas.com/rnd/datavisualization/

Another source of SAS/GRAPH examples and the SAS code used to produce them is...

SAS/GRAPH Software: Examples, Version 6, First Edition

http://www.sas.com/apps/pubscat/bookdetails.jsp?catid=1&pc=56022

An earlier use of the CALGRID style of data presentation can be found in the paper...

Mintz D, Fitz-Simons T, and Wayland M (1997), "Tracking Air Quality Trends with SAS/GRAPH®," *Proceedings of the Twenty-Second Annual SAS Users Group International Conference*, 22, 807-812.

Finally, an excellent resource for both selecting colors and obtaining their RGB hex values for use with SAS/GRAPH is...

http://www.colorbrewer.com

TRADEMARK CITATION

CONTACT INFORMATION

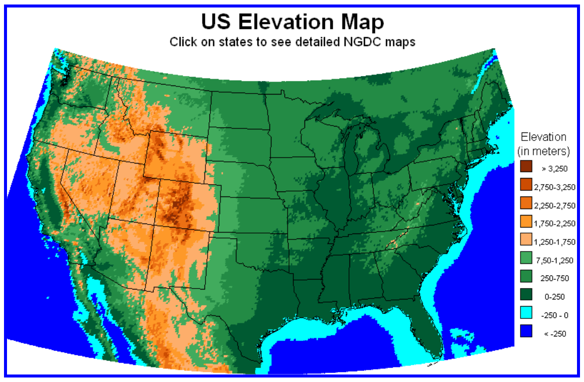
Either of the authors can be contacted using e-mail...

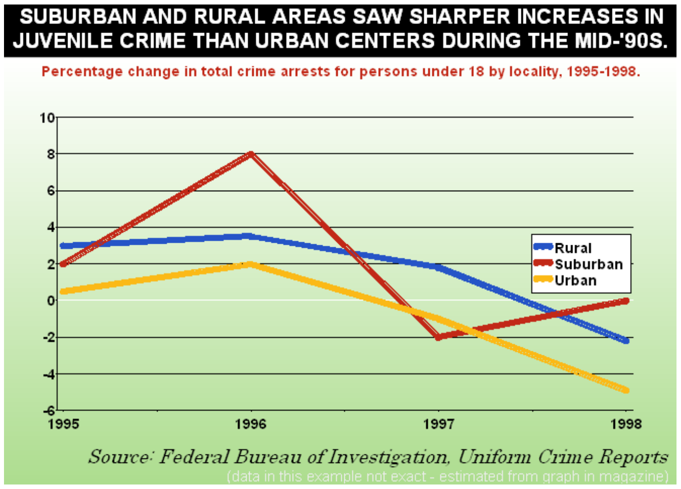
Either of the authors can be contacted using e-mail...	Mike Zdeb	msz03@albany.edu
	Robert Allison	Robert.Allison@sas.com

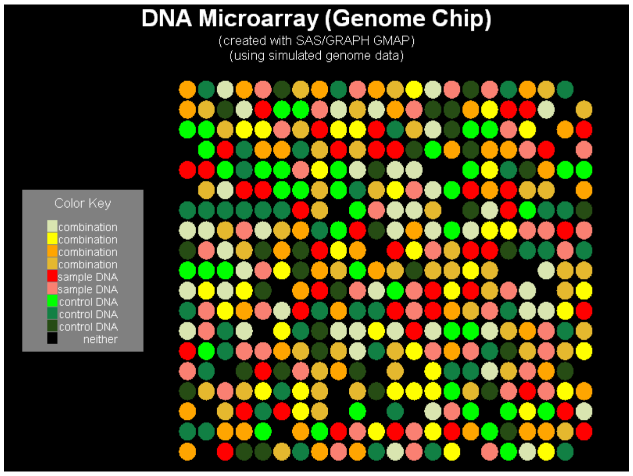
Send all requests for the SAS code used in this paper to Robert Allison.

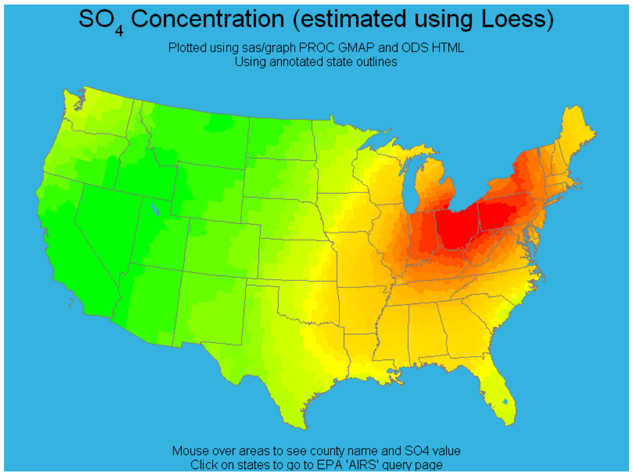
GRAPHICS GALLERY

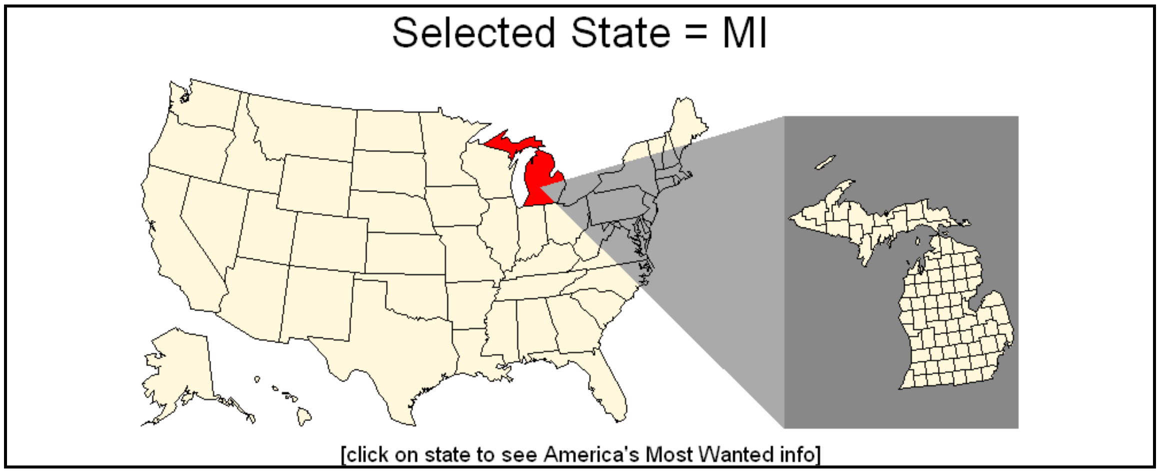
The following is a collection of SAS/GRAPH produced graphics. They all fall under the description '...wow, I did not know that you could do that with SAS/GRAPH...' Hopefully, the examples in this paper and this collection of graphics will increase your interest in learning more about the capabilities of SAS/GRAPH.

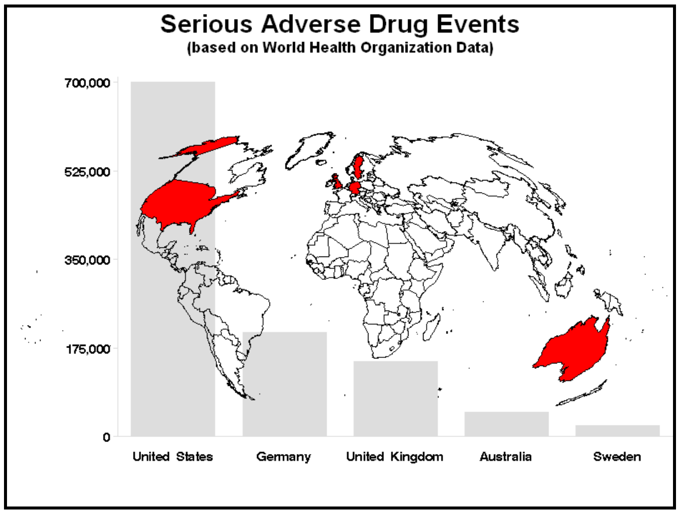


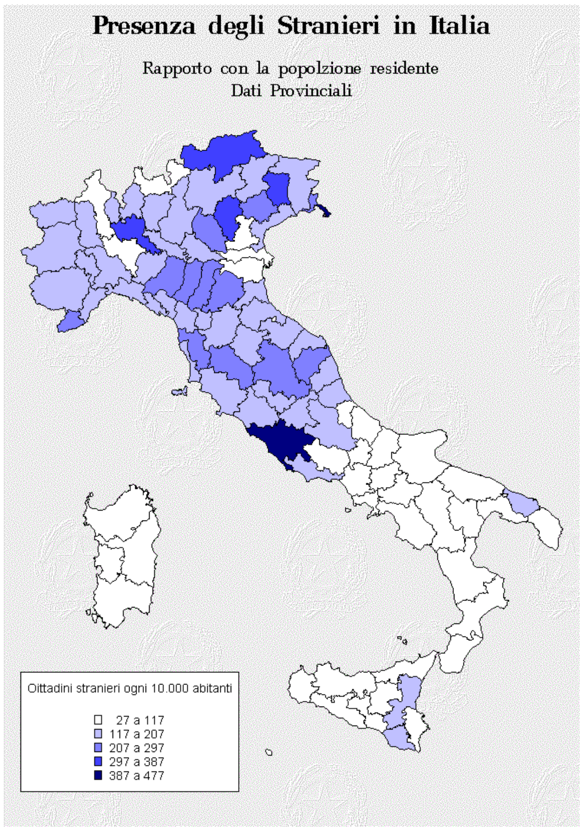


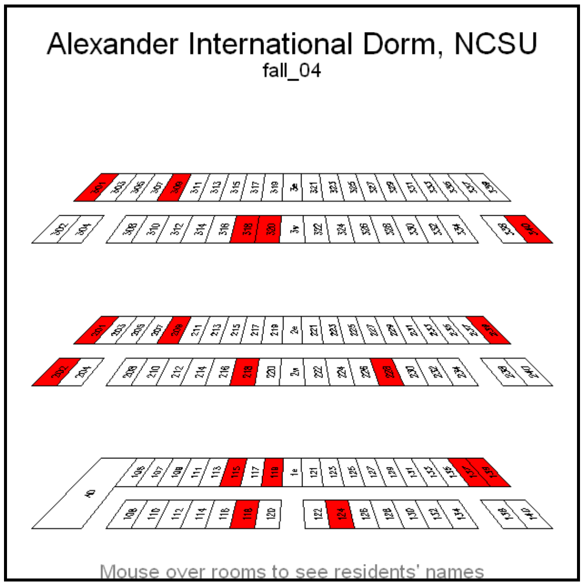


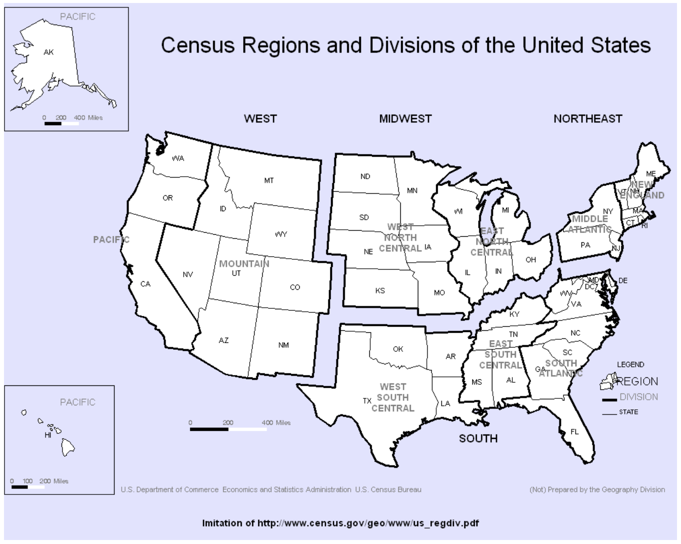

SO4 Concentration (estimated using Loess)
Plotted using sas/graph PROC GMAP and ODS HTML
Using annotated state outlines
Mouse over areas to see county name and SO4 value
Click on states to go to EPA 'AIRS' query page

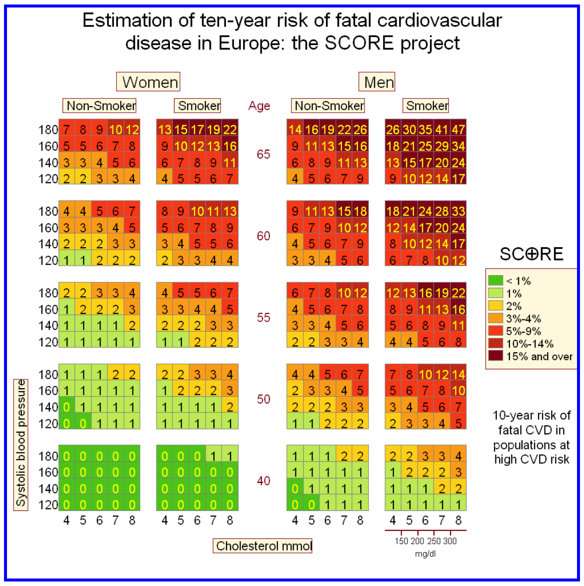





Presenza degli Stranieri in Italia
Rapporto con la popolzione residente
Dati Provinciali
Oittadini stranieri ogni 10.000 abitanti
27 a 117
117 a 207
207 a 297
297 a 387
387 a 477


Alexander International Dorm, NCSU
fall_04
Mouse over rooms to see residents' names


Census Regions and Divisions of the United States
PACIFIC
WEST
MIDWEST
NORTHEAST
SOUTH
MOUNTAIN
WEST NORTH CENTRAL
EAST NORTH CENTRAL
NEW ENGLAND
MIDDLE ATLANTIC
SOUTH ATLANTIC
EAST SOUTH CENTRAL
WEST SOUTH CENTRAL
LEGEND
REGION
DIVISION
STATE
U.S. Department of Commerce Economics and Statistics Administration U.S. Census Bureau
(Not) Prepared by the Geography Division
Imitation of http://www.census.gov/geo/www/us_regdiv.pdf



Estimation of ten-year risk of fatal cardiovascular disease in Europe: the SCORE project

Age	Systolic blood pressure	Women Non-Smoker 4	5	6	7	8	Women Smoker 4	5	6	7	8	Men Non-Smoker 4	5	6	7	8	Men Smoker 4	5	6	7	8
65	180	7	8	9	10	12	13	15	17	19	22	14	16	19	22	26	26	30	35	41	47
	160	5	5	6	7	8	9	10	12	13	16	9	11	13	15	16	18	21	25	29	34
	140	3	3	4	5	6	6	7	8	9	11	6	8	9	11	13	13	15	17	20	24
	120	2	2	3	3	4	4	5	5	6	7	4	5	6	7	9	9	10	12	14	17
60	180	4	4	5	6	7	8	9	10	11	13	9	11	13	15	18	18	21	24	28	33
	160	3	3	3	4	5	5	6	7	8	9	6	7	9	10	12	12	14	17	20	24
	140	2	2	2	3	3	3	4	5	5	6	4	5	6	7	9	8	10	12	14	17
	120	1	1	2	2	2	2	3	3	4	4	3	3	4	5	6	6	7	8	10	12
55	180	2	2	3	3	4	4	5	5	6	7	6	7	8	10	12	12	13	16	19	22
	160	1	2	2	2	3	3	3	4	4	5	4	5	6	7	8	8	9	11	13	16
	140	1	1	1	1	2	2	2	2	3	3	3	3	4	5	6	5	6	8	9	11
	120	1	1	1	1	1	1	1	2	2	2	2	2	3	3	4	4	4	5	6	8
50	180	1	1	1	2	2	2	2	3	3	4	4	4	5	6	7	7	8	10	12	14
	160	1	1	1	1	1	1	2	2	2	3	2	3	3	4	5	5	6	7	8	10
	140	0	1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	4	5	6	7
	120	0	0	1	1	1	1	1	1	1	1	1	1	2	2	2	2	3	3	4	5
40	180	0	0	0	0	0	0	0	0	1	1	1	1	1	2	2	2	2	3	3	4
	160	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	3
	140	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2
	120	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Cholesterol mmol

150 200 250 300 mg/dl

SC⊕RE

🟩	< 1%
🟩	1%
🟨	2%
🟧	3%-4%
🟥	5%-9%
🟥	10%-14%
🟥	15% and over

10-year risk of fatal CVD in populations at high CVD risk