Week 6: graphics

The main procedure we'll use is SGPLOT. I was able to use both GPLOT and SGPLOT in SAS Studio. SGPLOT has the advantage in SAS Studio that it takes less space. On my system at least, I have to maximize the screen to see everything in GPLOT.

SGPLOT was introduced with version 9.2 and is newer than GPLOT and seems to be intended to replace it. Years ago, GPLOT was an improvement over PLOT, which had purely text-based graphics. Many SAS programs have used GPLOT, so if you see examples of SAS code online or in textbooks, it is likely to use GPLOT instead of SGPLOT.

The SG in SGPLOT stands for Statistical Graphics.

Graphics

Some differences in GPLOT and SGPLOT include

1. Default of having rotated y-axis labels in SGPLOT (Default in GPLOT is for the y-axis label to appear at the upper left and non-rotated)
2. Graphics from SGPLOT can be produced in standard formats (PNG, JPEG, PDF) instead of SAS/GRAPH
3. GOPTIONS was used in GPLOT to control appearence of graphs, but not in SGPLOT.

SGPLOT: Source: Paper 154-2010, SAS Global Forum 2010, Using PROC SGPLOT for Quick High-Quality Graphs, Susan J. Slaughter, Avocet Solutions, Davis, CA Lora D. Delwiche, University of California, Davis, CA

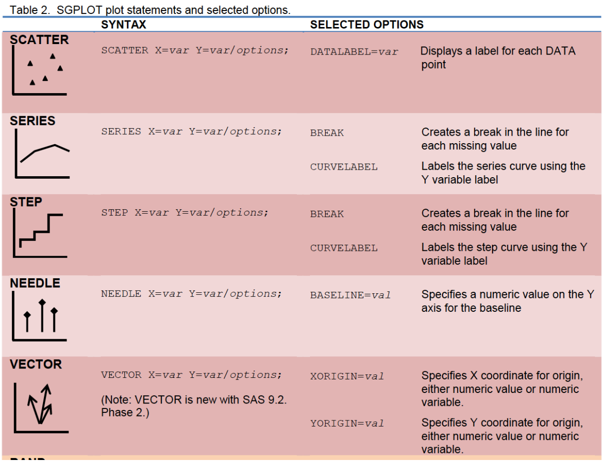
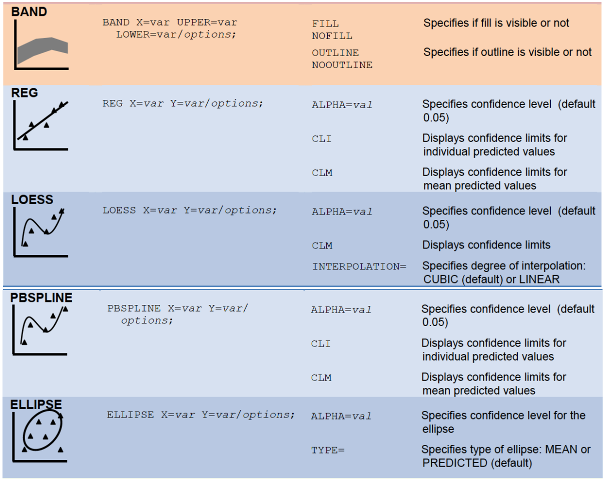


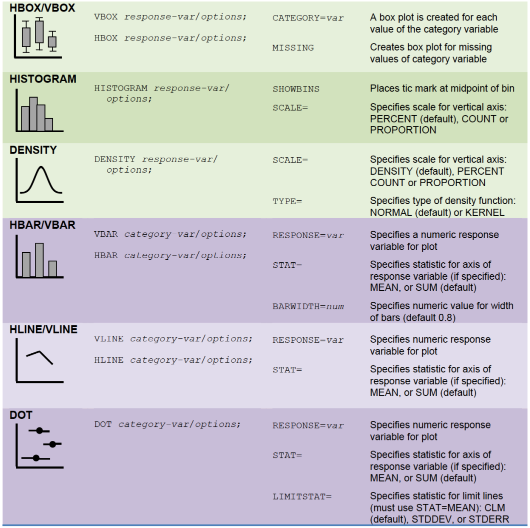
Table 2. SGPLOT plot statements and selected options.

	SYNTAX	SELECTED OPTIONS	
SCATTER	`SCATTER X=var Y=var/options;`	`DATALABEL=var`	Displays a label for each DATA point
SERIES	`SERIES X=var Y=var/options;`	`BREAK`	Creates a break in the line for each missing value
		`CURVELABEL`	Labels the series curve using the Y variable label
STEP	`STEP X=var Y=var/options;`	`BREAK`	Creates a break in the line for each missing value
		`CURVELABEL`	Labels the step curve using the Y variable label
NEEDLE	`NEEDLE X=var Y=var/options;`	`BASELINE=val`	Specifies a numeric value on the Y axis for the baseline
VECTOR	`VECTOR X=var Y=var/options;` (Note: VECTOR is new with SAS 9.2. Phase 2.)	`XORIGIN=val`	Specifies X coordinate for origin, either numeric value or numeric variable.
		`YORIGIN=val`	Specifies Y coordinate for origin, either numeric value or numeric variable.

SGPLOT



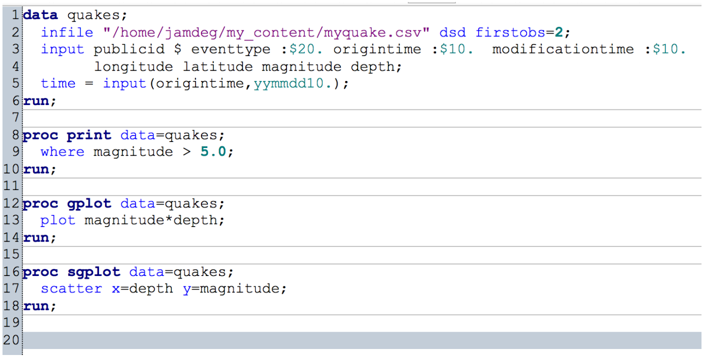
BAND	`BAND X=var UPPER=var LOWER=var/options;`	`FILL` `NOFILL`	Specifies if fill is visible or not
		`OUTLINE` `NOOUTLINE`	Specifies if outline is visible or not
REG	`REG X=var Y=var/options;`	`ALPHA=val`	Specifies confidence level (default 0.05)
		`CLI`	Displays confidence limits for individual predicted values
		`CLM`	Displays confidence limits for mean predicted values
LOESS	`LOESS X=var Y=var/options;`	`ALPHA=val`	Specifies confidence level (default 0.05)
		`CLM`	Displays confidence limits
		`INTERPOLATION=`	Specifies degree of interpolation: CUBIC (default) or LINEAR
PBSPLINE	`PBSPLINE X=var Y=var/ options;`	`ALPHA=val`	Specifies confidence level (default 0.05)
		`CLI`	Displays confidence limits for individual predicted values
		`CLM`	Displays confidence limits for mean predicted values
ELLIPSE	`ELLIPSE X=var Y=var/options;`	`ALPHA=val`	Specifies confidence level for the ellipse
		`TYPE=`	Specifies type of ellipse: MEAN or PREDICTED (default)



HBOX/VBOX	`VBOX response-var/options;` `HBOX response-var/options;`	`CATEGORY=var`	A box plot is created for each value of the category variable
		`MISSING`	Creates box plot for missing values of category variable
HISTOGRAM	`HISTOGRAM response-var/ options;`	`SHOWBINS`	Places tic mark at midpoint of bin
		`SCALE=`	Specifies scale for vertical axis: PERCENT (default), COUNT or PROPORTION
DENSITY	`DENSITY response-var/ options;`	`SCALE=`	Specifies scale for vertical axis: DENSITY (default), PERCENT COUNT or PROPORTION
		`TYPE=`	Specifies type of density function: NORMAL (default) or KERNEL
HBAR/VBAR	`VBAR category-var/options;` `HBAR category-var/options;`	`RESPONSE=var`	Specifies a numeric response variable for plot
		`STAT=`	Specifies statistic for axis of response variable (if specified): MEAN, or SUM (default)
		`BARWIDTH=num`	Specifies numeric value for width of bars (default 0.8)
HLINE/VLINE	`VLINE category-var/options;` `HLINE category-var/options;`	`RESPONSE=var`	Specifies numeric response variable for plot
		`STAT=`	Specifies statistic for axis of response variable (if specified): MEAN, or SUM (default)
DOT	`DOT category-var/options;`	`RESPONSE=var`	Specifies numeric response variable for plot
		`STAT=`	Specifies statistic for axis of response variable (if specified): MEAN, or SUM (default)
		`LIMITSTAT=`	Specifies statistic for limit lines (must use STAT=MEAN): CLM (default), STDDEV, or STDERR

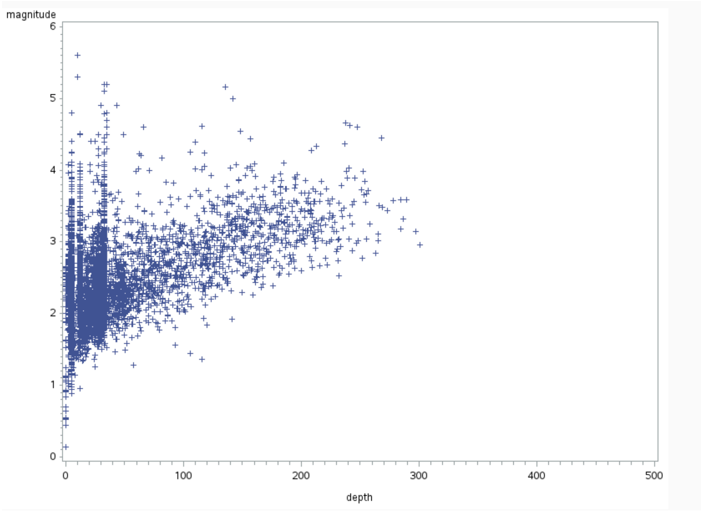
GPLOT and SGPlot

The most common type of plot in statistics is perhaps the scatterplot. Here's how to do it in both GPLOT and SGPLOT using default options.



```
data quakes;
  infile "/home/jamdeg/my_content/myquake.csv" dsd firstobs=2;
  input publicid $ eventtype :$20. origintime :$10.  modificationtime :$10.
        longitude latitude magnitude depth;
  time = input(origintime,yymmdd10.);
run;

proc print data=quakes;
  where magnitude > 5.0;
run;

proc gplot data=quakes;
  plot magnitude*depth;
run;

proc sgplot data=quakes;
  scatter x=depth y=magnitude;
run;

```

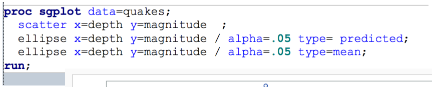
GPLOT



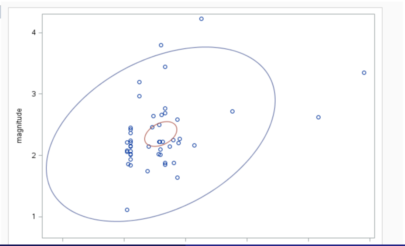
SGPLOT: scatterplot

SGPLOT: scatterplot with elliptical prediction and confidence bands

This is typical for how options work in SAS procedures—after a forward slash, you put a list of options. For the ellipse, the prediction ellipse is the default so doesn't need to be specified (but this makes your code easier to follow if you type it). Note that I reduced the number of observations to 50 so that the confidence ellipse for the mean would be visible.

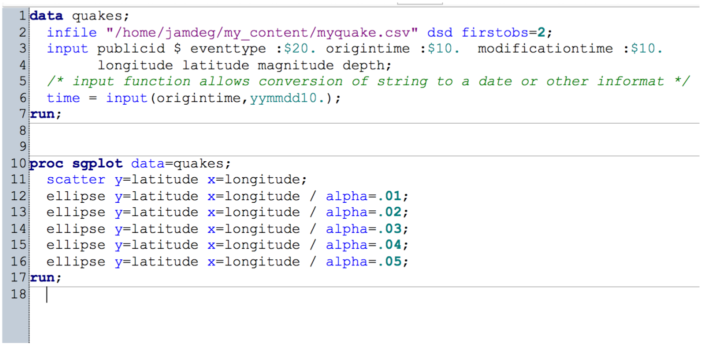


```
proc sgplot data=quakes;
  scatter x=depth y=magnitude  ;
  ellipse x=depth y=magnitude / alpha=.05 type= predicted;
  ellipse x=depth y=magnitude / alpha=.05 type=mean;
run;
```



SGPLOT: scatterplot with elliptical prediction interval

You can put as many prediction ellipses as you want, for example with different α levels. Note that you can put the x and y coordinates in either order.

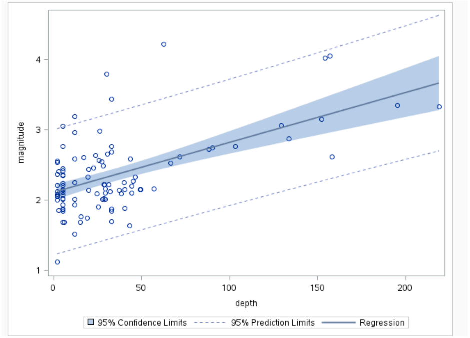


```
data quakes;
  infile "/home/jamdeg/my_content/myquake.csv" dsd firstobs=2;
  input publicid $ eventtype :$20. origintime :$10.  modificationtime :$10.
        longitude latitude magnitude depth;
  /* input function allows conversion of string to a date or other informat */
  time = input(origintime,yymmdd10.);
run;


proc sgplot data=quakes;
  scatter y=latitude x=longitude;
  ellipse y=latitude x=longitude / alpha=.01;
  ellipse y=latitude x=longitude / alpha=.02;
  ellipse y=latitude x=longitude / alpha=.03;
  ellipse y=latitude x=longitude / alpha=.04;
  ellipse y=latitude x=longitude / alpha=.05;
run;
```

SGPLOT: scatterplot with regression line

You can use SGPLOT to do simple linear regression instead of using PROC

```
proc sgplot data=quake3 (obs=100);
  reg x=depth y=magnitude / cli clm;
run;
```



Tips on color in graphs

A good rule of thumb is to try to use color that will print well in black-in-white or greyscale. This is because (1) some of your audience might be color blind, (2) if your presentation or article or paper is printed, it is likely to be printed in black-and-white, and you want it to be legible.

For these reasons, I'm likely to use say, blue and orange instead of blue and red. Or, orange and red instead of green and red. Orange is lighter than red and will be distinguishable in greyscale printing.

Another tip is to use plot symbols in addition to color, or dotted lines in addition to solid lines.

Tips on color in graphs

Thin lines are harder to distinguish colors than thicker lines, as are hollow symbols (hollow circles instead of solid).

Another example that can be difficult is in heat maps, where intensity is mapped to a rainbow with say, red being large values and blue being low values. Although these maps are pretty, it can be difficult to interpret either in black and white or in color.
The most common form of color-blindness is red-green, which effects about 8% of men and 5% of women of Norther European descent (according to Wikipedia). The percentage is lower in Asians (4% of men) and Africans (5% of men). A rule of thumb from genetics is that if 1/20 men is affected, 1/400 women would be. It is likely that in a classroom or a lecture at a conference, at least one audience member is color-blind in some form.

Tips on color in graphs

Here's an example of a bad choice of color in my opinion (from a 2008 SAS Global Forum, by Susan Schwartz, a SAS employee). The <\$30,000 and \$40-\$50,000 categories look essentially the same to me (probably one is red and one is green?). There also could be a gradient going from light (less expensive cars) to dark (more expensive), but instead the lightest shade is in the middle. This will not be ideal in black and white.

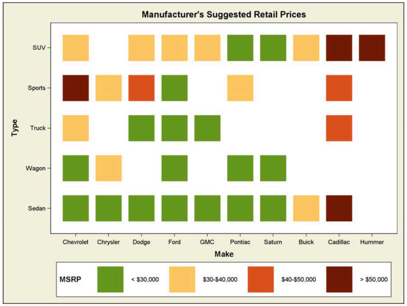
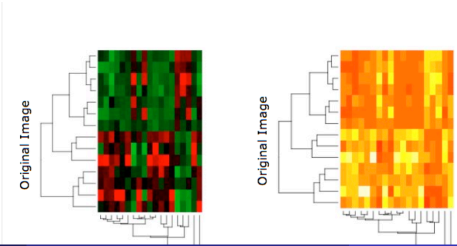


Figure 8: Heatmap using the SGPLOT procedure

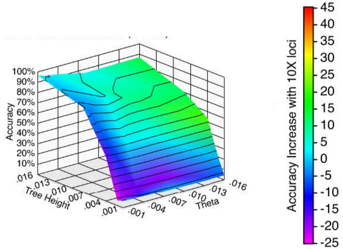
Tips on color in graphs: heatmaps

This is an exmaple of heatmap of gene expression levels. Red-green heatmaps are very common but hard for many people to read and again won't print we'll in black in white. A white-yellow-orange-red color scheme



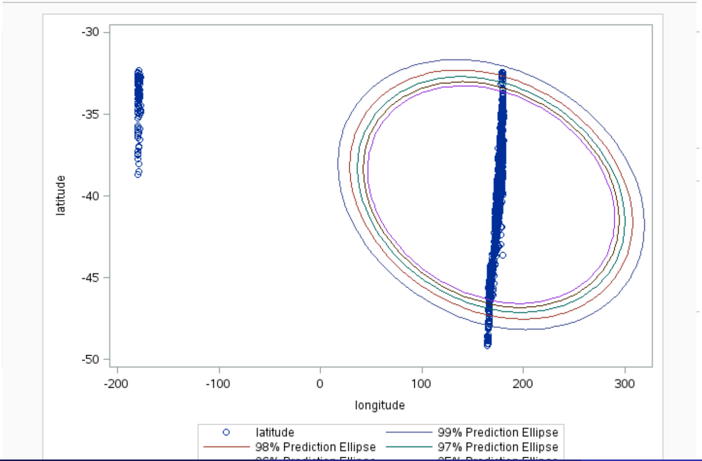
Tips on color in graphs: heatmaps

The following example is from a research paper in my area. Although I liked the paper, the heat map, I cannot distinguish 0% from -25% (blue versus purple) as well as +35% and 10% (red versus green). The 4-d plot is pretty (dimensions are theta, tree height, percent accuracy, and change in accuracy), but would be much easier to read with white-yellow-orange-red color scheme.



SGPLOT: scatterplot with elliptical prediction interval

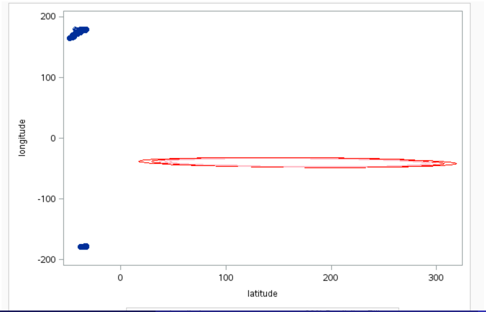
SGPLOT automatically makes a nice legend, although the colors are hard for to decipher.



SGPLOT: scatterplot with elliptical prediction interval

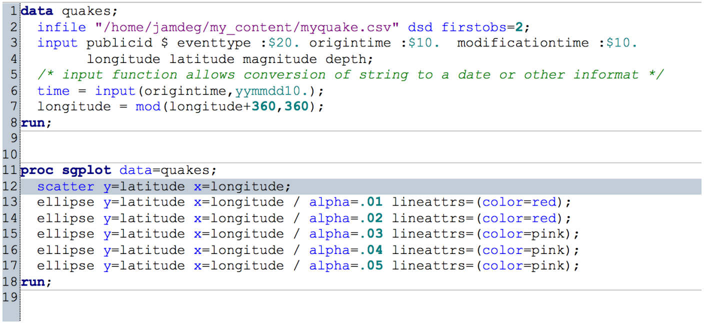
Notice that if you accidentally swap x and y, it does not generate an error, but might make a nonsensical graph.

```
proc sgplot data=quakes;
  scatter x=latitude y=longitude;
  ellipse y=latitude x=longitude / alpha=.01 lineattrs=(color=red);
  ellipse y=latitude x=longitude / alpha=.02 lineattrs=(color=red);
  ellipse y=latitude x=longitude / alpha=.03 lineattrs=(color=pink);
  ellipse y=latitude x=longitude / alpha=.04 lineattrs=(color=pink);
  ellipse y=latitude x=longitude / alpha=.05 lineattrs=(color=pink);
run;
```



SGPLOT: scatterplot with elliptical prediction interval

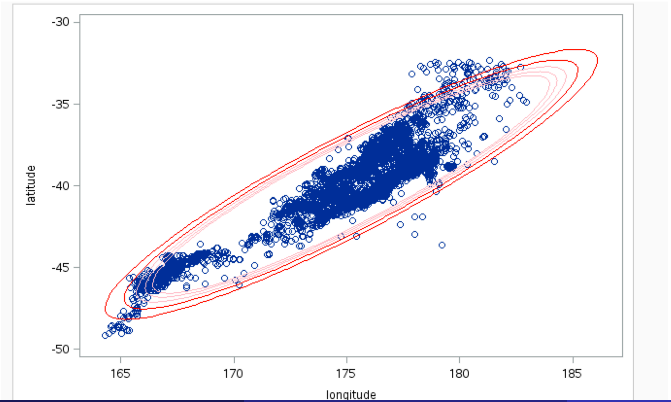
Let's fix the negative longitudes using the modulus function again and see what happens.



```
data quakes;
  infile "/home/jamdeg/my_content/myquake.csv" dsd firstobs=2;
  input publicid $ eventtype :$20. origintime :$10.  modificationtime :$10.
        longitude latitude magnitude depth;
  /* input function allows conversion of string to a date or other informat */
  time = input(origintime,yymmdd10.);
  longitude = mod(longitude+360,360);
run;


proc sgplot data=quakes;
  scatter y=latitude x=longitude;
  ellipse y=latitude x=longitude / alpha=.01 lineattrs=(color=red);
  ellipse y=latitude x=longitude / alpha=.02 lineattrs=(color=red);
  ellipse y=latitude x=longitude / alpha=.03 lineattrs=(color=pink);
  ellipse y=latitude x=longitude / alpha=.04 lineattrs=(color=pink);
  ellipse y=latitude x=longitude / alpha=.05 lineattrs=(color=pink);
run;

```

SGPLOT: scatterplot with elliptical prediction interval

Interpretation of prediction interval should be that if observations come from the same process, then $(1-\alpha)\%$ of future observations should come from the the ellipse corresponding to α. Does that apply in this case? Why or why not?

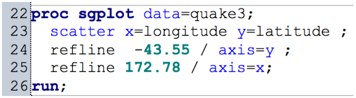


SGPLOT: scatterplot with elliptical prediction interval

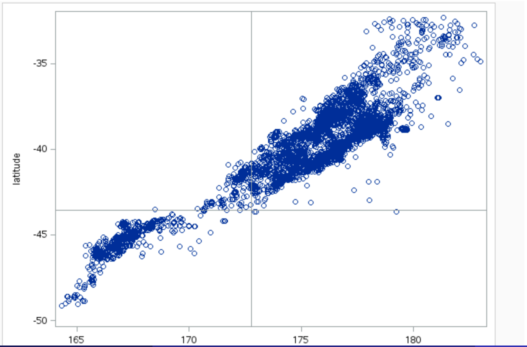
You can also use other line-attributes using the LINEATTRS option, such as line-type (i.e., dotted or dashed) and thickness. Mostly, you will have to search online to find various options for customizing your plot. We will encounter some more commonly used options along the way.

SGPLOT: scatterplot with reference lines

Here I added longitudinal and latitudinal references lines for my previous residence of Christchurch, New Zealand.



```
proc sgplot data=quake3;
  scatter x=longitude y=latitude ;
  refline  -43.55 / axis=y ;
  refline 172.78 / axis=x;
run;
```

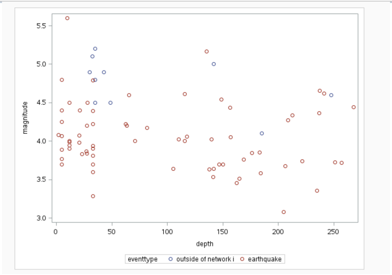


SGPLOT: bivariate plot with two groups

You can also have a grouping variable. The default behavior seems to depend on global options in your system, and doesn do what I would like in SAS Studio, which is to have distinct symbols as well as distinct colors.

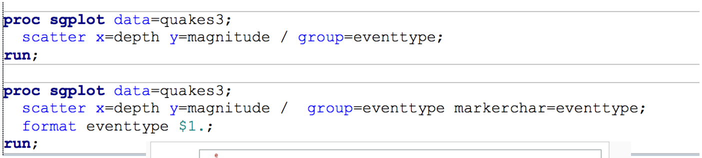
```
proc sgplot data=quakes3;
  scatter x=depth y=magnitude / group=eventtype;
run;
```

```
proc sgplot data=quakes3;
  scatter x=depth y=magnitude /  group=eventtype markerchar=eventtype;
  format eventtype $1.;
run;
```

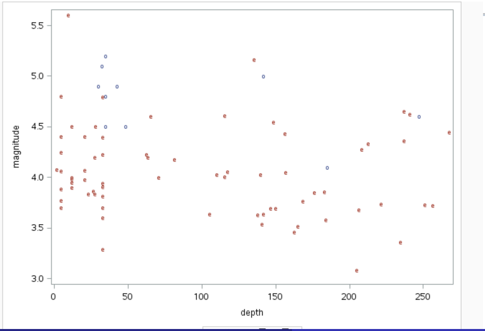


SGPLOT: bivariate plot with two groups

In this case, I used the value of the variable as the marker for the variable. To make it less cluttered, I abbreviated it to one character using a format

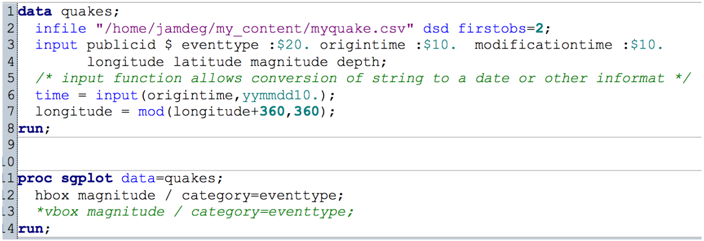


```
proc sgplot data=quakes3;
  scatter x=depth y=magnitude / group=eventtype;
run;

proc sgplot data=quakes3;
  scatter x=depth y=magnitude /  group=eventtype markerchar=eventtype;
  format eventtype $1.;
run;
```



SGPLOT: boxplots

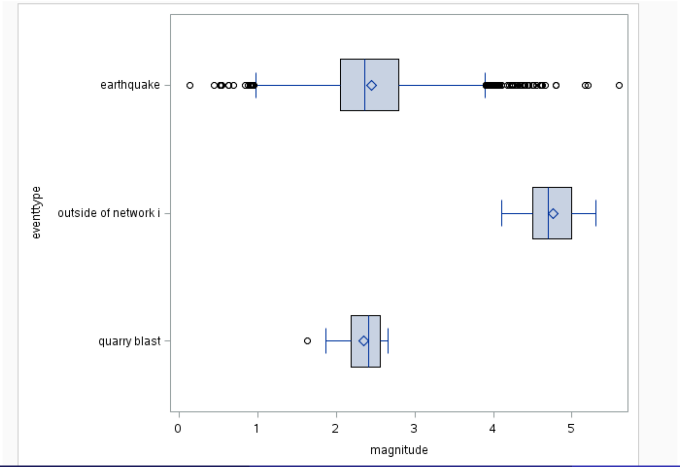
Another common type of plot is boxplots. This can be done either vertically or horizontally. For example,



```
data quakes;
  infile "/home/jamdeg/my_content/myquake.csv" dsd firstobs=2;
  input publicid $ eventtype :$20. origintime :$10.  modificationtime :$10.
        longitude latitude magnitude depth;
  /* input function allows conversion of string to a date or other informat */
  time = input(origintime,yymmdd10.);
  longitude = mod(longitude+360,360);
run;


proc sgplot data=quakes;
  hbox magnitude / category=eventtype;
  *vbox magnitude / category=eventtype;
run;
```

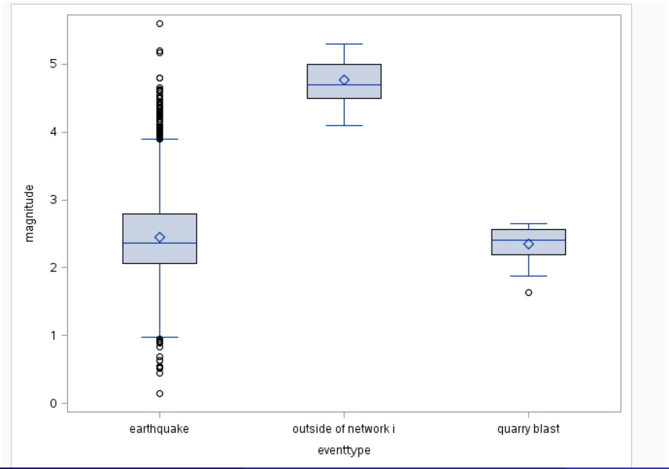
SGPLOT: boxplots

Another common type of plot is boxplots. This can be done either vertically or horizontally. For example,



SGPLOT: boxplots

Another common type of plot is boxplots. This can be done either vertically or horizontally. For example,

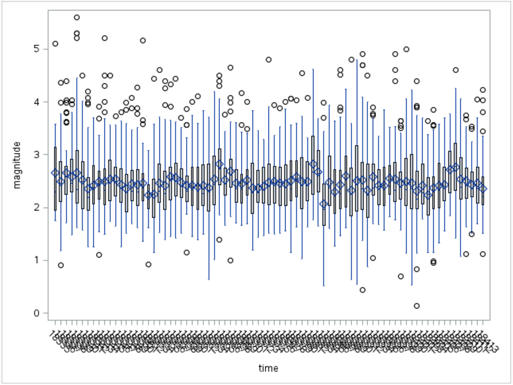


SGPLOT: boxplots

There are many options (over two dozen) for boxplots, such as the width, whether or not outliers are surpressed, whether or not the median is surpressed, how the whiskers are defined, color, etc. Here we look at boxplots of magnitude for every day in the data set. This would be a quick way of seeing if there are trends in the distribution over time, which there don't appear to be here. Using SAS date functions, you could instead use weeks of the year, or month or other units of time.

SGPLOT: boxplots

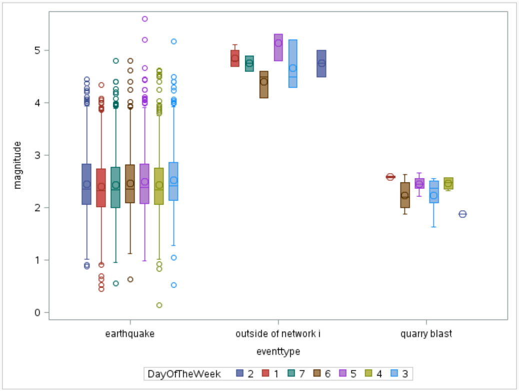
```
proc sgplot data=quakes;
  *hbox magnitude / category=eventtype;
  vbox magnitude / category=time;

run;
```



SGPLOT: boxplots

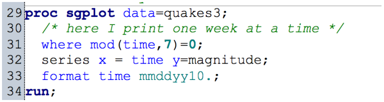
Boxplots with two grouping variables

```
proc sgplot data=quake3;
  vbox magnitude / category=eventtype group=DayOfTheWeek;
run;
```

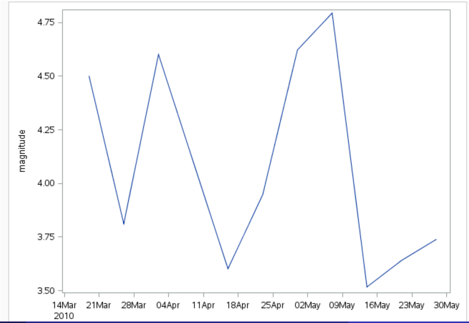


SGPLOT: lineplots, series plots

Here I created a data set that consists of days and the largest magnitude quake on that day. The series plots these as a time series, but only plotting one day per week. Note the format statement so that the *x*-axis is



```
proc sgplot data=quakes3;
  /* here I print one week at a time */
  where mod(time,7)=0;
  series x = time y=magnitude;
  format time mmddyy10.;
run;
```



What day of the week was in different years?

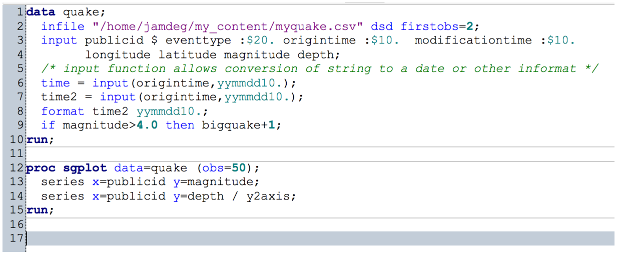
The previous example used days of the week that were the same day of the week as January 1st, 1960. Since the modulus is 7, this means that the number of days of the week since January 1st, 1960 is a multiple of 7. If we had used `where mod(time,7)=3` instead, then we'd be using Mondays. How to tell the day of the week for January 1st, 1960?

One way is to go online, for example to http://calendarhome.com/print-a-calendar/. Or you can use the linux function `cal` where you type `$ cal 2010` to get the full calendar for the year 2010.

You can also use the WEEKDAY function in SAS, which converts a date to a day, 1 through 7, where 1=Sunday, and 7=Saturday.

SGPLOT: lineplots, series plots

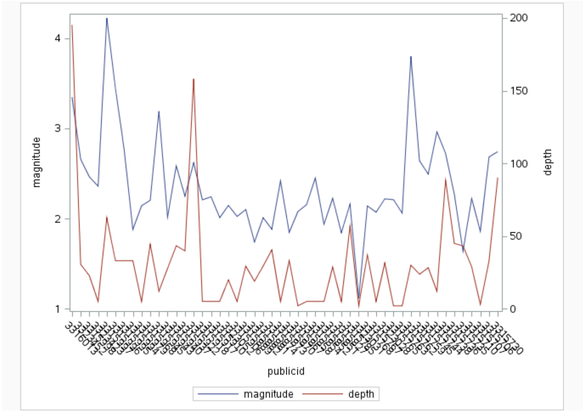
You can also superimpose multiple lineplots and create a secondary axis.



```
data quake;
  infile "/home/jamdeg/my_content/myquake.csv" dsd firstobs=2;
  input publicid $ eventtype :$20. origintime :$10.  modificationtime :$10.
        longitude latitude magnitude depth;
  /* input function allows conversion of string to a date or other informat */
  time = input(origintime,yymmdd10.);
  time2 = input(origintime,yymmdd10.);
  format time2 yymmdd10.;
  if magnitude>4.0 then bigquake+1;
run;

proc sgplot data=quake (obs=50);
  series x=publicid y=magnitude;
  series x=publicid y=depth / y2axis;
run;

```

SGPLOT: lineplots, series plots

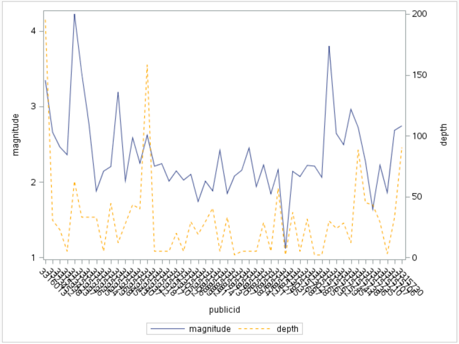
You can also superimpose multiple lineplots and create a secondary axis.



SGPLOT: lineplots, series plots

Options to modify line type and color.

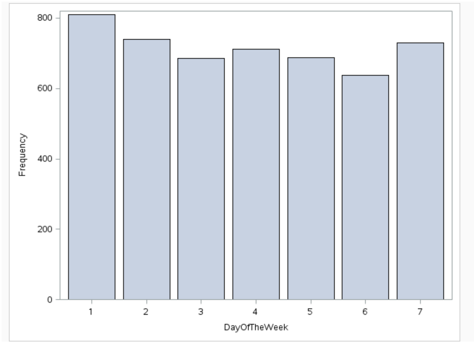
```
proc sgplot data=quake (obs=50);
  series x=publicid y=magnitude;
  series x=publicid y=depth / y2axis lineattrs=(pattern=2 color=orange);
run;
```



SGPLOT: bar plots

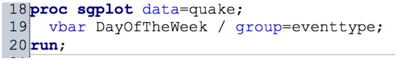
Basic bar plots are easy to do with SGPLOT. Similarly to boxplots, they can be either vertical (VBAR) or horizontal (HBAR).

```
proc sgplot data=quakes;
    vbar DayOfTheWeek;
run;
```

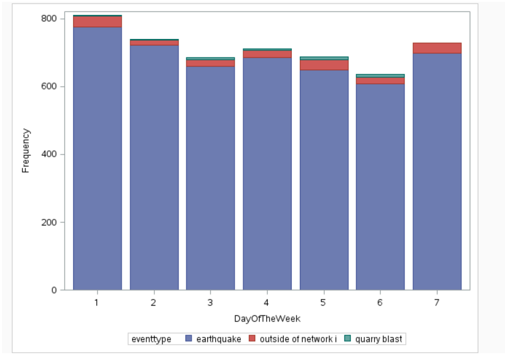


SGPLOT: bar plots

Stacked barplots.



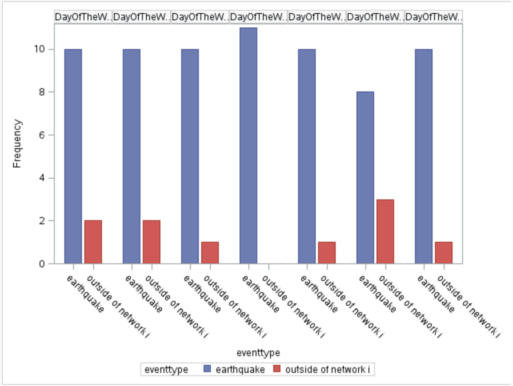
```
proc sgplot data=quake;
  vbar DayOfTheWeek / group=eventtype;
run;
```



SGPLOT: bar plots (well SGPANEL actually)

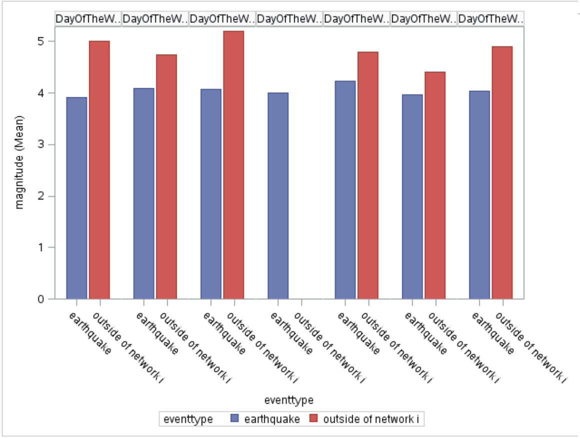
This allows another variable in your barplot.

```
proc sgpanel data=quakes3;
panelby DayOfTheWeek / layout=columnlattice onepanel noborder;
vbar eventtype / group=eventtype ;
run;
```

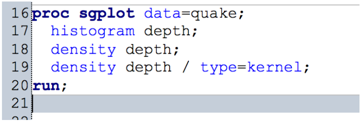


SGPLOT: bar plots (well SGPANEL actually)

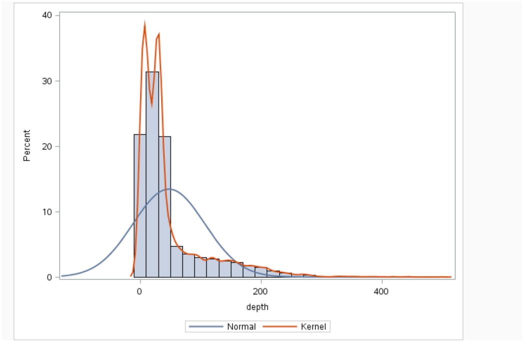
```
proc sgpanel data=quakes3;
panelby DayOfTheWeek / layout=columnlattice onepanel noborder;
vbar eventtype / response = magnitude stat=mean group=eventtype ;
run;
```



SGPLOT: histograms



```
proc sgplot data=quake;
  histogram depth;
  density depth;
  density depth / type=kernel;
run;
```



SGPLOT: histogram kernel density estimate

The idea behind the kernel density estimate is that the height of the curve is

$$\widehat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

where h is the bandwidth (a chosen parameter), and $x_1, \dots, n$ are the data. Usually $K()$ is small for values far from 0, so that the height of the curve near x is determined mostly by data near x. Data far from x contribute less to value of the function near x. A Gaussian kernel for example will treat the argument to $K()$ as being like a z-score, so values more extreme than ± 3 will have negligible contribution.

The function $\widehat{f}$ estimates the density f, and will be increasingly accurate as $n \to \infty$ and $h \to 0$ assuming that the data are i.i.d.

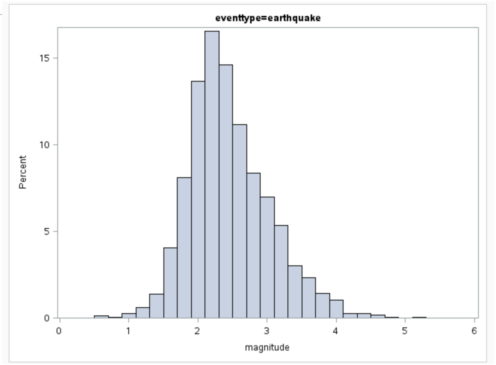
Using BY statments with SAS plots

A common way of generating plots in SAS is to use a BY statement with a (usually) categorical variable. This creates a separate plot for each value of the BY variable, and can of course end up generating LOTS of output.

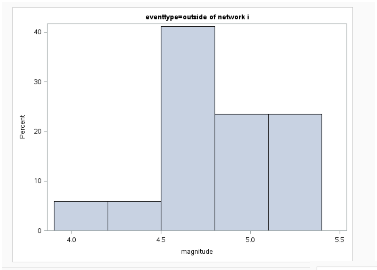
As usual, use of the BY statement requires that the data be sorted on the BY variables first.

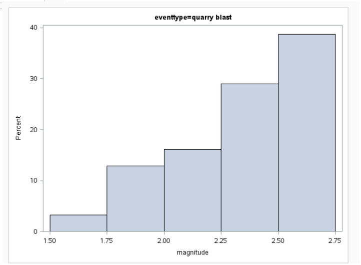
SGPLOT: histograms

```
proc sort data=quake out=quake3;
 by eventtype;
run;

proc sgplot data=quake3;
  by eventtype;
  histogram magnitude;
run;
```

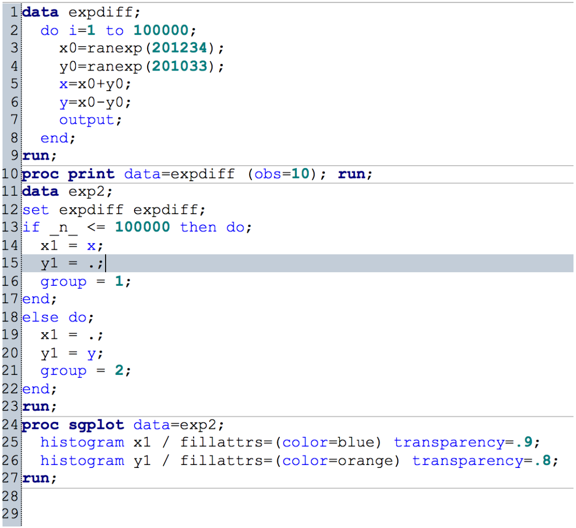


SGPLOT: histograms



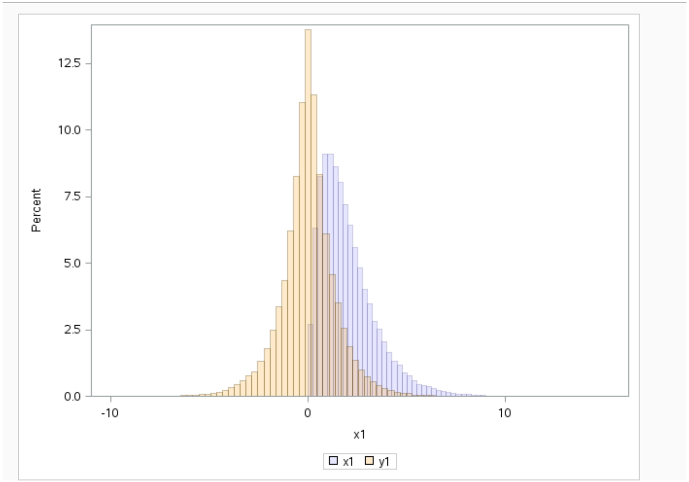


Overlapping histograms



```
data expdiff;
  do i=1 to 100000;
    x0=ranexp(201234);
    y0=ranexp(201033);
    x=x0+y0;
    y=x0-y0;
    output;
  end;
run;
proc print data=expdiff (obs=10); run;
data exp2;
set expdiff expdiff;
if _n_ <= 100000 then do;
  x1 = x;
  y1 = .;
  group = 1;
end;
else do;
  x1 = .;
  y1 = y;
  group = 2;
end;
run;
proc sgplot data=exp2;
  histogram x1 / fillattrs=(color=blue) transparency=.9;
  histogram y1 / fillattrs=(color=orange) transparency=.8;
run;


```

Overlapping histograms

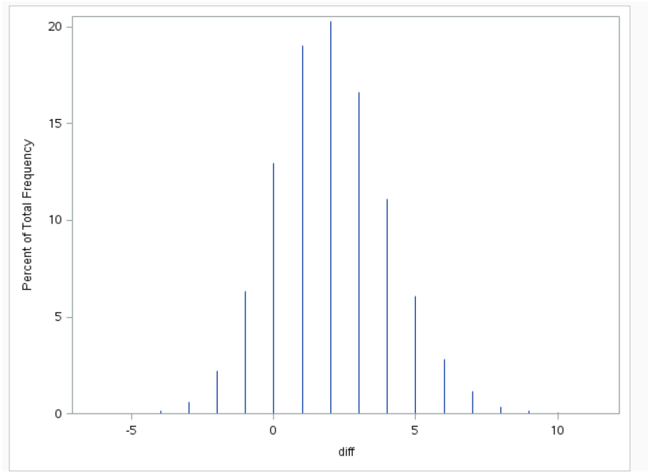


SGPLOT: needle plots

I like using needle plots for discrete probability distributions. Here I simulated $X \sim \text{Poiss}(1)$ and $Y \sim \text{Poiss}(3)$ and plotted the distribution of $Y - X$. Note that the distribution includes negative values.

```
data poissdiff;
  do i=1 to 100000;
    x=ranpoi(201234,1);
    y=ranpoi(201033,3);
    diff = y-x;
    output;
  end;
run;

proc print data=poissdiff (obs=10);
run;

proc sort data=poissdiff;
by diff;
run;

proc freq data=poissdiff;
  tables diff / out=freqout;
run;

proc sgplot data=freqout;
  needle x=diff y=percent;
run;
```

Other ways to customize graphs



Still to come...

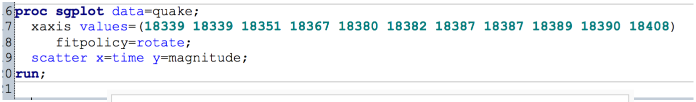
Other things we'd like to do with graphs include

1. control tick marks (customize, log scale etc)
2. control font sizes (tricky with SGPLOT)
3. control which color schemes are used
4. control which plot symbols are used
5. place legends in graph if desired
6. Have titles/labels with different fonts, subscripts, etc.
7. Use panels for subgraphs
8. 3d graphs

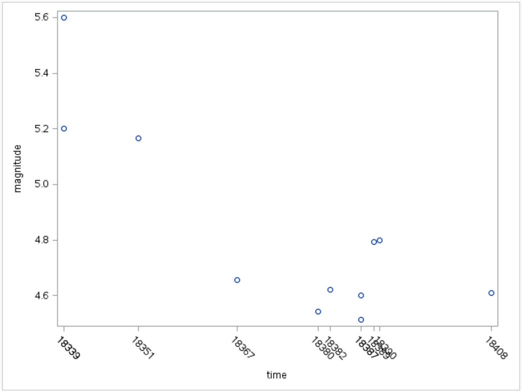
Custom tick marks on axes

Here we give an example where we want irregular spacing on an axis. Suppose I want to plot earthquakes with magntitude greater than 5.0, and I want to display the dates on the x-axis that correspond to the data instead of making them regularly spaced.

Irregular tick marks

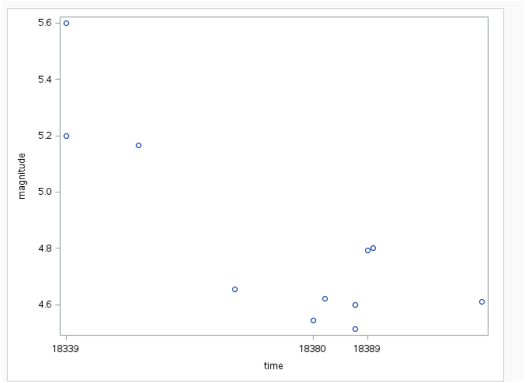


```
proc sgplot data=quake;
  xaxis values=(18339 18339 18351 18367 18380 18382 18387 18387 18389 18390 18408)
      fitpolicy=rotate;
  scatter x=time y=magnitude;
run;
```



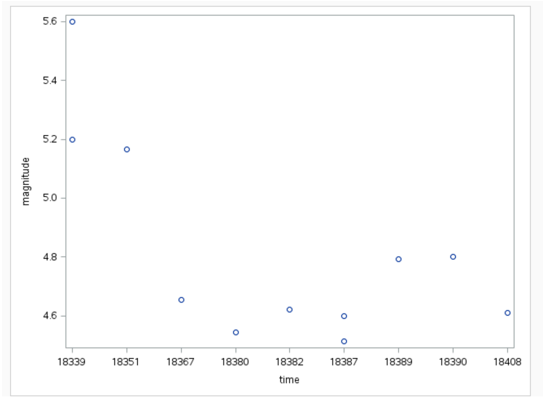
Irregular tick marks

```
proc sgplot data=quake;
  xaxis values=(18339 18339 18351 18367 18380 18382 18387 18387 18389 18390 18408);
  scatter x=time y=magnitude;
run;
```



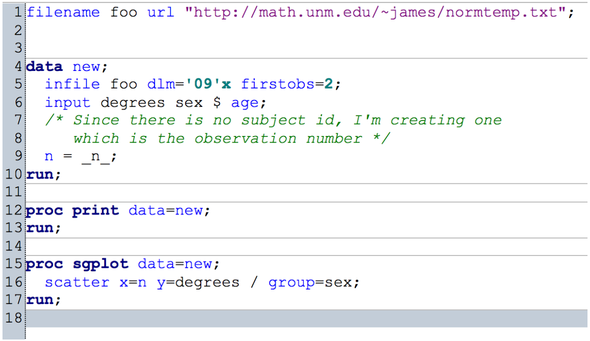
Irregular tick marks

```
proc sgplot data=quake;
  xaxis type=discrete values=(18339 18339 18351 18367 18380 18382 18387 18387 18389 18390 18408);
  scatter x=time y=magnitude;
run;
```



Example temperature data

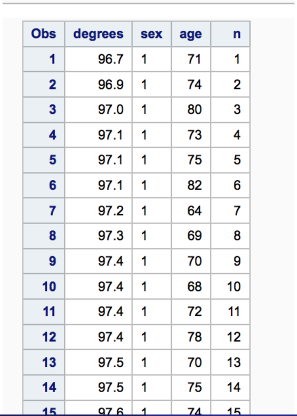
We'll start with the temperature data which is here sorted by sex and temperature. We'll look at various ways we can modify (hopefully improve)



```
filename foo url "http://math.unm.edu/~james/normtemp.txt";


data new;
  infile foo dlm='09'x firstobs=2;
  input degrees sex $ age;
  /* Since there is no subject id, I'm creating one
     which is the observation number */
  n = _n_;
run;

proc print data=new;
run;

proc sgplot data=new;
  scatter x=n y=degrees / group=sex;
run;

```

Temperature data: Using a grouping variable

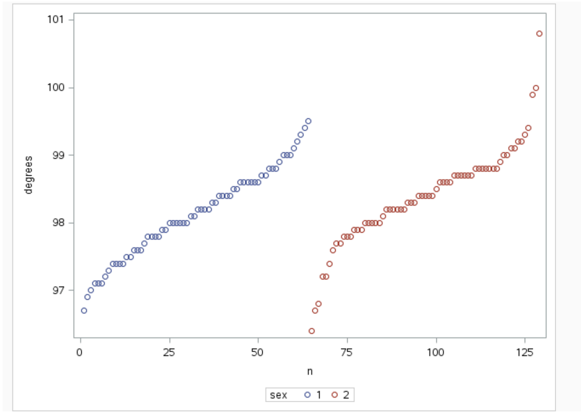
We'll start with the temperature data which is already sorted by sex and temperature. Here 1=man, 2=woman. Notice that we added the *n* variable so that we have something like a subject ID.



Obs	degrees	sex	age	n
1	96.7	1	71	1
2	96.9	1	74	2
3	97.0	1	80	3
4	97.1	1	73	4
5	97.1	1	75	5
6	97.1	1	82	6
7	97.2	1	64	7
8	97.3	1	69	8
9	97.4	1	70	9
10	97.4	1	68	10
11	97.4	1	72	11
12	97.4	1	78	12
13	97.5	1	70	13
14	97.5	1	75	14
15	97.6	1	74	15

Temperature data

We'll start with the temperature data which is already sorted by sex and temperature. Here 1=man, 2=woman.

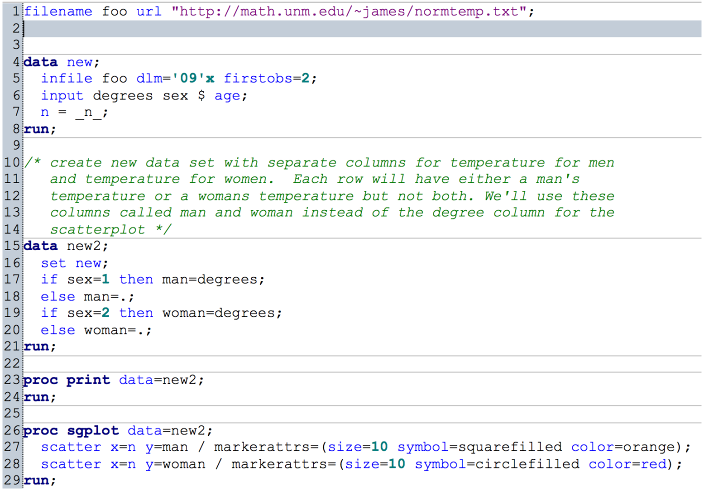


Temperature data

We'd like to improve the temperature plot in several ways. These include making the plot symbols filled rather than hollow, making the plotting characters bigger, and having distinct plot symbols for the two sexes.

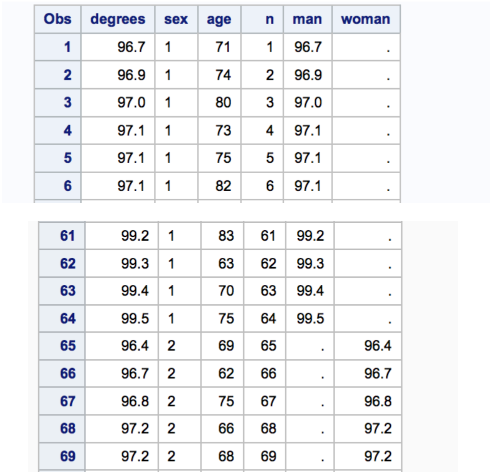
Unfortunately, it is a little bit tricky to do this. To accomplish this, I will create two columns for temperature–one for men and one for women. Then I will make a scatterplot of each separately and overlay them on top of each other. By doing this, I can control the appearance of each one separately.

Temperature data



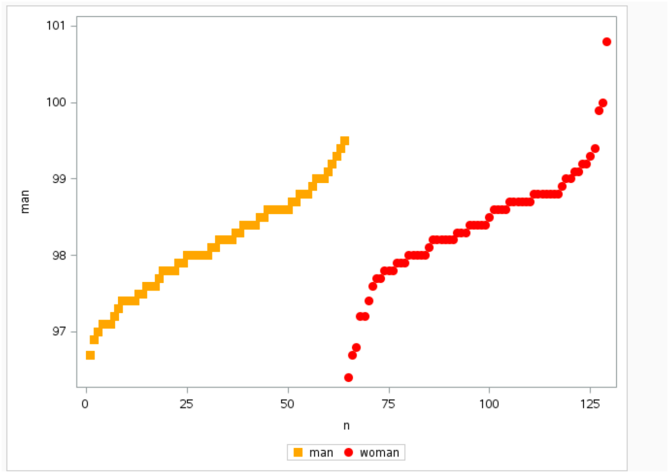
```
filename foo url "http://math.unm.edu/~james/normtemp.txt";


data new;
  infile foo dlm='09'x firstobs=2;
  input degrees sex $ age;
  n = _n_;
run;

/* create new data set with separate columns for temperature for men
   and temperature for women.  Each row will have either a man's
   temperature or a womans temperature but not both. We'll use these
   columns called man and woman instead of the degree column for the
   scatterplot */
data new2;
  set new;
  if sex=1 then man=degrees;
  else man=.;
  if sex=2 then woman=degrees;
  else woman=.;
run;

proc print data=new2;
run;

proc sgplot data=new2;
  scatter x=n y=man / markerattrs=(size=10 symbol=squarefilled color=orange);
  scatter x=n y=woman / markerattrs=(size=10 symbol=circlefilled color=red);
run;
```

Temperature data



Obs	degrees	sex	age	n	man	woman
1	96.7	1	71	1	96.7	.
2	96.9	1	74	2	96.9	.
3	97.0	1	80	3	97.0	.
4	97.1	1	73	4	97.1	.
5	97.1	1	75	5	97.1	.
6	97.1	1	82	6	97.1	.
61	99.2	1	83	61	99.2	.
62	99.3	1	63	62	99.3	.
63	99.4	1	70	63	99.4	.
64	99.5	1	75	64	99.5	.
65	96.4	2	69	65	.	96.4
66	96.7	2	62	66	.	96.7
67	96.8	2	75	67	.	96.8
68	97.2	2	66	68	.	97.2
69	97.2	2	68	69	.	97.2

Temperature data



Temperature data

How can we improve the plot at this point?
Here are some possibilities:

1. Change the *y*-axis label. Note that the y-axis says "man" which is the name of the first *y* variable used, but is not an appropriate name for the *y*-axis
2. Add a second axis with a Celcius scale
3. Change the axis tick marks on the *x*-axis so that the indexing starts at 1 for the data on women
4. Add a reference line and/or tick mark at 98.6 degrees Farenheight, considered to be the average.

Temperature data

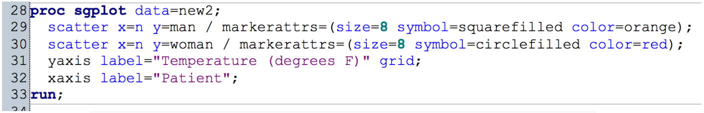
We might think about other ways of visualizing this data. We could plot all of the data together using age on the x-axis rather than sex, but still have color and symbols to differentiate them.

We could also plot overlapping histograms of the two distributions.

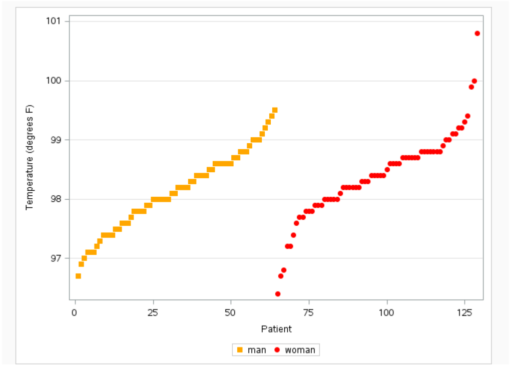
If we wanted to do a more formal test of whether the two distributions agree, we could plot their empirical distribution functions simultaneously and use a Kolmogorov-Smirnov (K-S) test.

FInally we could do something like a Q-Q plot, plotting sorted men's temperatures against sorted women's temperatures, but this is a little tricky if the sample sizes are different. If the sample sizes are the same and the distributions are the same, this plot should roughly follow a straight line.

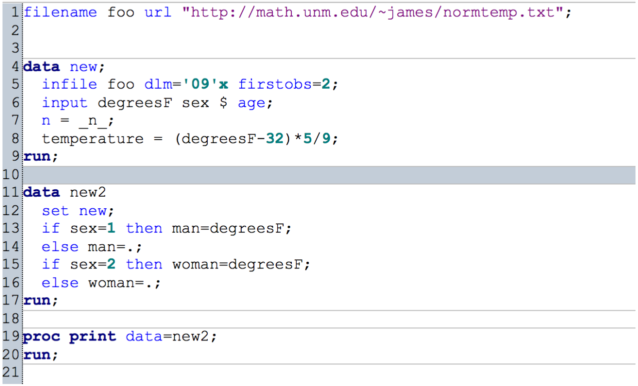
Temperature data: Fixing the labels and adding a grid



```
proc sgplot data=new2;
  scatter x=n y=man / markerattrs=(size=8 symbol=squarefilled color=orange);
  scatter x=n y=woman / markerattrs=(size=8 symbol=circlefilled color=red);
  yaxis label="Temperature (degrees F)" grid;
  xaxis label="Patient";
run;
```

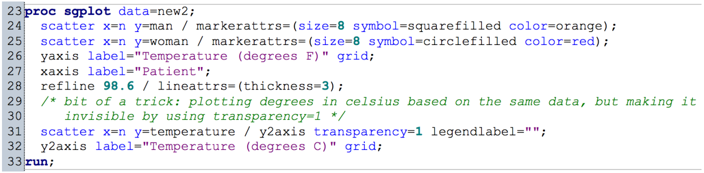


Temperature data: adding a Celsius scale



```
filename foo url "http://math.unm.edu/~james/normtemp.txt";


data new;
  infile foo dlm='09'x firstobs=2;
  input degreesF sex $ age;
  n = _n_;
  temperature = (degreesF-32)*5/9;
run;

data new2
  set new;
  if sex=1 then man=degreesF;
  else man=.;
  if sex=2 then woman=degreesF;
  else woman=.;
run;

proc print data=new2;
run;

```

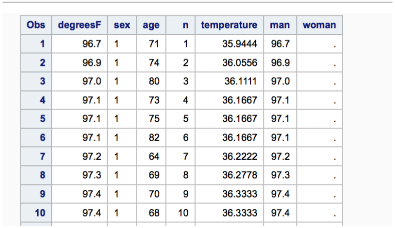
Temperature data: adding a Celsius scale



```sas
proc sgplot data=new2;
  scatter x=n y=man / markerattrs=(size=8 symbol=squarefilled color=orange);
  scatter x=n y=woman / markerattrs=(size=8 symbol=circlefilled color=red);
  yaxis label="Temperature (degrees F)" grid;
  xaxis label="Patient";
  refline 98.6 / lineattrs=(thickness=3);
  /* bit of a trick: plotting degrees in celsius based on the same data, but making it
     invisible by using transparency=1 */
  scatter x=n y=temperature / y2axis transparency=1 legendlabel="";
  y2axis label="Temperature (degrees C)" grid;
run;
```

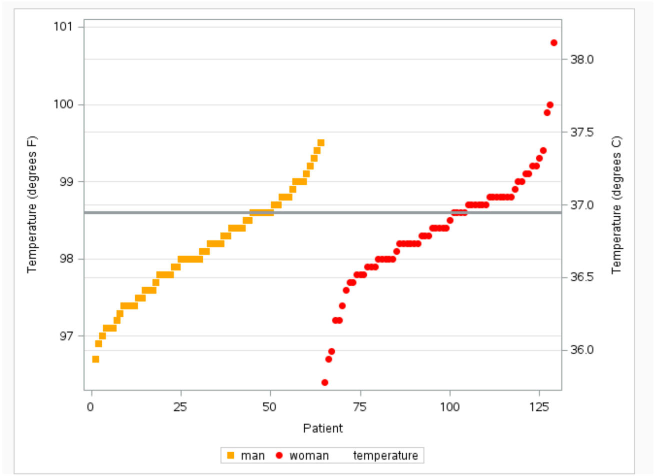
Temperature data: adding a Celsius scale

```
proc sgplot data=new2;
  scatter x=n y=man / markerattrs=(size=8 symbol=squarefilled color=orange);
  scatter x=n y=woman / markerattrs=(size=8 symbol=circlefilled color=red);
  yaxis label="Temperature (degrees F)" grid;
  xaxis label="Patient";
  refline 98.6 / lineattrs=(thickness=3);
  /* bit of a trick: plotting degrees in celsius based on the same data, but making it
     invisible by using transparency=1 */
  scatter x=n y=temperature / y2axis transparency=1 legendlabel="";
  y2axis label="Temperature (degrees C)" grid;
run;
```

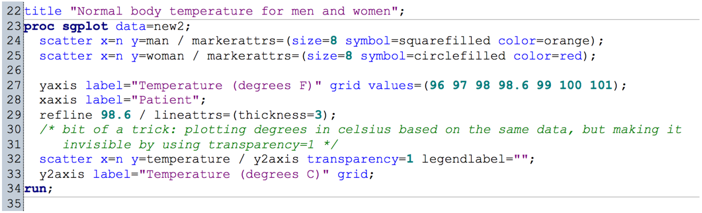


Obs	degreesF	sex	age	n	temperature	man	woman
1	96.7	1	71	1	35.9444	96.7	.
2	96.9	1	74	2	36.0556	96.9	.
3	97.0	1	80	3	36.1111	97.0	.
4	97.1	1	73	4	36.1667	97.1	.
5	97.1	1	75	5	36.1667	97.1	.
6	97.1	1	82	6	36.1667	97.1	.
7	97.2	1	64	7	36.2222	97.2	.
8	97.3	1	69	8	36.2778	97.3	.
9	97.4	1	70	9	36.3333	97.4	.
10	97.4	1	68	10	36.3333	97.4	.

Temperature data: adding a Celsius scale

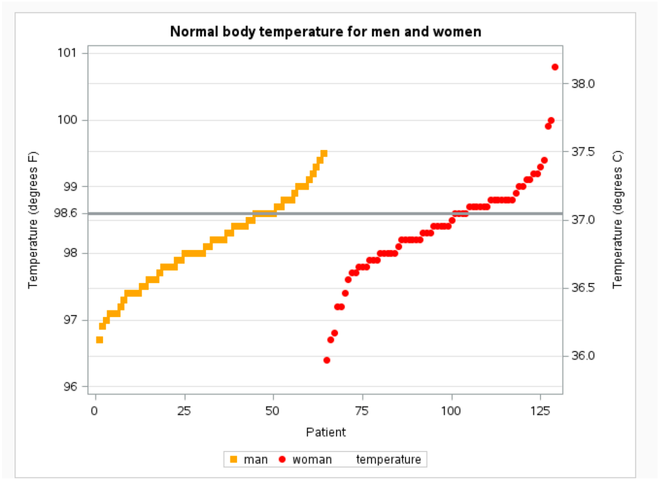


Temperature data: adding a Celsius scale

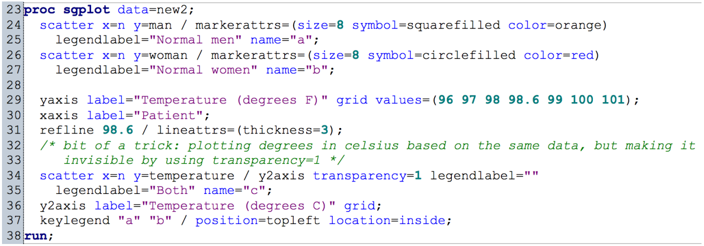


```
title "Normal body temperature for men and women";
proc sgplot data=new2;
  scatter x=n y=man / markerattrs=(size=8 symbol=squarefilled color=orange);
  scatter x=n y=woman / markerattrs=(size=8 symbol=circlefilled color=red);

  yaxis label="Temperature (degrees F)" grid values=(96 97 98 98.6 99 100 101);
  xaxis label="Patient";
  refline 98.6 / lineattrs=(thickness=3);
  /* bit of a trick: plotting degrees in celsius based on the same data, but making it
     invisible by using transparency=1 */
  scatter x=n y=temperature / y2axis transparency=1 legendlabel="";
  y2axis label="Temperature (degrees C)" grid;
run;
```

Temperature data: adding a Celsius scale



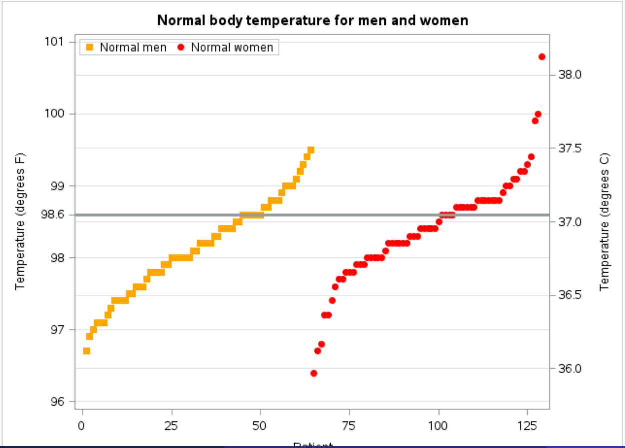
Temperature data: adding a Celsius scale



```
proc sgplot data=new2;
  scatter x=n y=man / markerattrs=(size=8 symbol=squarefilled color=orange)
    legendlabel="Normal men" name="a";
  scatter x=n y=woman / markerattrs=(size=8 symbol=circlefilled color=red)
    legendlabel="Normal women" name="b";

  yaxis label="Temperature (degrees F)" grid values=(96 97 98 98.6 99 100 101);
  xaxis label="Patient";
  refline 98.6 / lineattrs=(thickness=3);
  /* bit of a trick: plotting degrees in celsius based on the same data, but making it
     invisible by using transparency=1 */
  scatter x=n y=temperature / y2axis transparency=1 legendlabel=""
    legendlabel="Both" name="c";
  y2axis label="Temperature (degrees C)" grid;
  keylegend "a" "b" / position=topleft location=inside;
run;
```

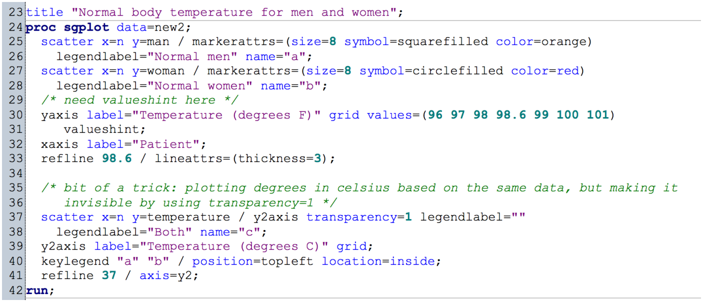
Temperature data: adding a Celsius scale

I've noticed a problem that the Farenheit and Celsius values don't match exactly because 98.6°F = 37.0°C.



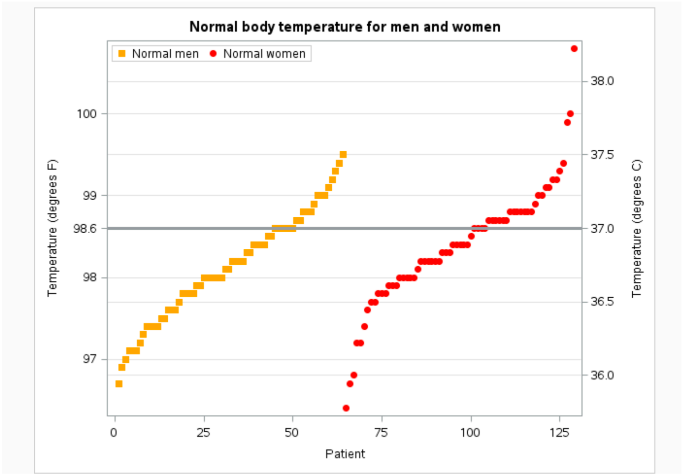
Temperature data: fixing range of plots with `valueshint`

This option makes the range of axis depend on the data rather than the values specified and fixes the previous problem.



```
title "Normal body temperature for men and women";
proc sgplot data=new2;
  scatter x=n y=man / markerattrs=(size=8 symbol=squarefilled color=orange)
    legendlabel="Normal men" name="a";
  scatter x=n y=woman / markerattrs=(size=8 symbol=circlefilled color=red)
    legendlabel="Normal women" name="b";
  /* need valueshint here */
  yaxis label="Temperature (degrees F)" grid values=(96 97 98 98.6 99 100 101)
     valueshint;
  xaxis label="Patient";
  refline 98.6 / lineattrs=(thickness=3);

  /* bit of a trick: plotting degrees in celsius based on the same data, but making it
     invisible by using transparency=1 */
  scatter x=n y=temperature / y2axis transparency=1 legendlabel=""
    legendlabel="Both" name="c";
  y2axis label="Temperature (degrees C)" grid;
  keylegend "a" "b" / position=topleft location=inside;
  refline 37 / axis=y2;
run;
```

Temperature data: fixing range of plots with `valueshint`



Increasing font size

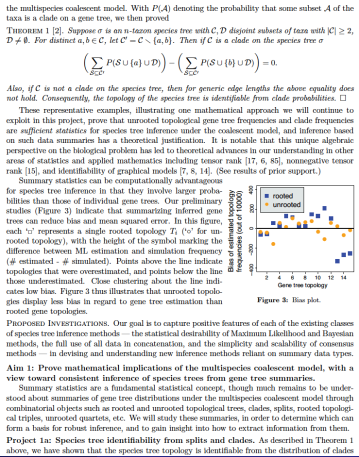
Font size requires working with the ODS (output delivery system) outside of PROC SGPLOT. The idea is to use PROC TEMPLATE to change the style settings to create a customized style, then use the ODS to say that you want to use this customized style when running SG procedures.

Unfortunately, attempting this in SAS Studio gives permissions erros, and I haven't gotten examples from the web to work in linux either (they create other errors). So I think we are stuck with font sizes as they come in SAS.

You can use ODS to output a pdf file of your graphic (in linux SAS – I'm not sure where it goes in SAS Studio), and this creates a graphic that is about 1/2 of a page (all the width and half the length).

Increasing font size

To give you an idea of why it is necessary to increase font sizes so often, I'll show you a recent example from a grant proposal.



the multispecies coalescent model. With $P(\mathcal{A})$ denoting the probability that some subset $\mathcal{A}$ of the taxa is a clade on a gene tree, we then proved

Theorem 1 [2]. *Suppose σ is an n-taxon species tree with $\mathcal{C}, \mathcal{D}$ disjoint subsets of taxa with $|\mathcal{C}| \geq 2$, $\mathcal{D} \neq \emptyset$. For distinct $a, b \in \mathcal{C}$, let $\mathcal{C}' = \mathcal{C} \setminus \{a, b\}$. Then if $\mathcal{C}$ is a clade on the species tree σ*

$$\left(\sum_{\mathcal{S} \subseteq \mathcal{C}'} P(\mathcal{S} \cup \{a\} \cup \mathcal{D}) \right) - \left(\sum_{\mathcal{S} \subseteq \mathcal{C}'} P(\mathcal{S} \cup \{b\} \cup \mathcal{D}) \right) = 0.$$

Also, if $\mathcal{C}$ is not a clade on the species tree, then for generic edge lengths the above equality does not hold. Consequently, the topology of the species tree is identifiable from clade probabilities. □

These representative examples, illustrating one mathematical approach we will continue to exploit in this project, prove that unrooted topological gene tree frequencies and clade frequencies are *sufficient statistics* for species tree inference under the coalescent model, and inference based on such data summaries has a theoretical justification. It is notable that this unique algebraic perspective on the biological problem has led to theoretical advances in our understanding in other areas of statistics and applied mathematics including tensor rank [17, 6, 85], nonnegative tensor rank [15], and identifiability of graphical models [7, 8, 14]. (See results of prior support.)

Summary statistics can be computationally advantageous for species tree inference in that they involve larger probabilities than those of individual gene trees. Our preliminary studies (Figure 3) indicate that summarizing inferred gene trees can reduce bias and mean squared error. In this figure, each '□' represents a single rooted topology T_i ('○' for unrooted topology), with the height of the symbol marking the difference between ML estimation and simulation frequency (# estimated - # simulated). Points above the line indicate topologies that were overestimated, and points below the line those underestimated. Close clustering about the line indicates low bias. Figure 3 thus illustrates that unrooted topologies display less bias in regard to gene tree estimation than rooted gene topologies.

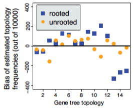


Figure 3: Bias plot.

Proposed Investigations. Our goal is to capture positive features of each of the existing classes of species tree inference methods — the statistical desirability of Maximum Likelihood and Bayesian methods, the full use of all data in concatenation, and the simplicity and scalability of consensus methods — in devising and understanding new inference methods reliant on summary data types.

Aim 1: Prove mathematical implications of the multispecies coalescent model, with a view toward consistent inference of species trees from gene tree summaries.

Summary statistics are a fundamental statistical concept, though much remains to be understood about summaries of gene tree distributions under the multispecies coalescent model through combinatorial objects such as rooted and unrooted topological trees, clades, splits, rooted topological triples, unrooted quartets, etc. We will study these summaries, in order to determine which can form a basis for robust inference, and to gain insight into how to extract information from them.

Project 1a: Species tree identifiability from splits and clades. As described in Theorem 1 above, we have shown that the species tree topology is identifiable from the distribution of clades

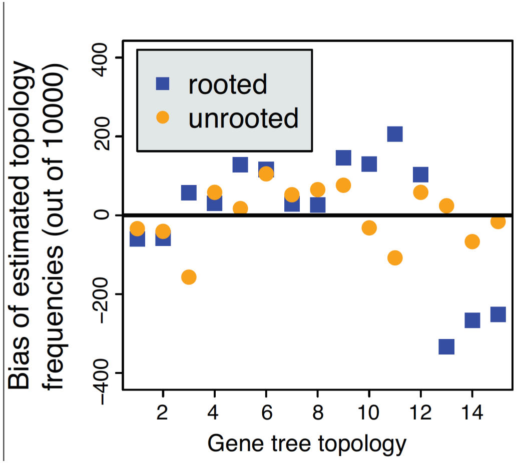
Increasing font size

The typical way to create figures is to create a full size figure, for example as pdf or better (especially for journal submissions), as encapsulated postscript files (`.eps`). Then the graphic is shrunk down to fit in the text. You'll get better resolution if you start with a large graphic and shrink it down than if you try to make the graphic the final intended size.

Notice that the font size on the axes of the plot are similar to the surrounding text. This means that if you looked at this graphic by opening the `.eps` file that it was from, the font size would look rather large.

Increasing font size

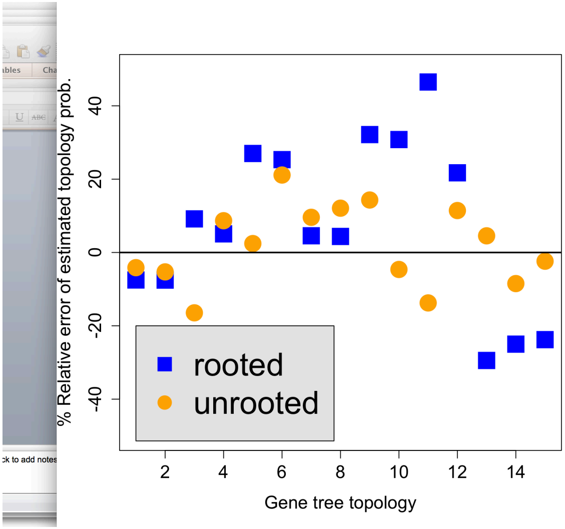
This figure was modified in Adobe Illustrator from a figure done in R.



Increasing font size

Although R is pretty good about allowing you to customize font sizes, it has its limits, and it is not easy to make y-axis labels as large as I often like, which is part of the reason this graphic was modified by Illustrator instead of using R directly. This screen shot shows how making the y-axis font puts the y-label right to edge of the screen.

Increasing font size

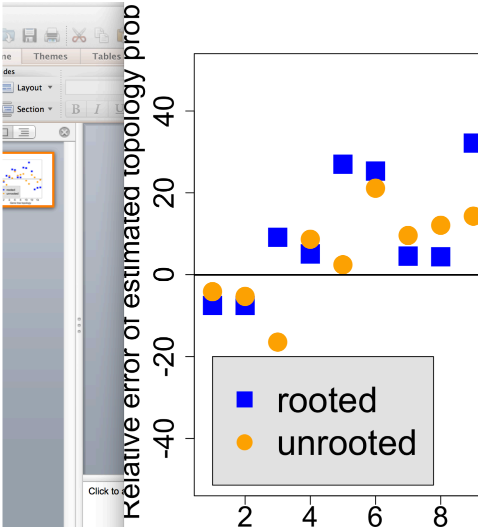


Increasing font size

The previous example used `cex=1.4`, where `cex` is a scaling factor in R that tells it how much to increase font sizes by. The data points used `cex=3`, and the legend used `cex=2.5` (as large as I could get without covering the data). The legend was shaded so that points in the legend did not appear to belong to the graph.

Now we'll take a look at what happens if I use `cex=2.0`

Increasing font size



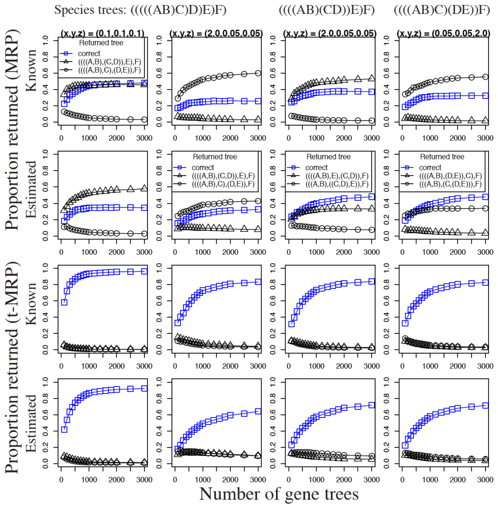
Increasing font size

The point is that increasing font sizes in output is a very common problem, and both SAS and R are disappointing in this regard. Here are some possible solutions that I have tried (with R):

1. Use Illustrator to fix things, including *y*-axis labels, tic labels, etc.
2. Be happy with `cex=1.4` for R. Also `mtext` (for marginal text) can help in R.
3. If really desperate in R, make an empty graph, plot your symbols inside with no axes, then plot the axes and tic marks as line segments so that the plotting area is smaller than usual. This gives you more space on the margins
4. Keep the tic marks and axis labels empty, read the figure into LaTeX, then annotate the graph in LaTeX. If you are using LaTeX, then this has the advantage that you can get fonts that match your text. This approach works but is very time-consuming and requires a lot of trial error to specify the coordinates in LaTeXcorrectly.

An example of the LaTeXapproach follows

Increasing font size



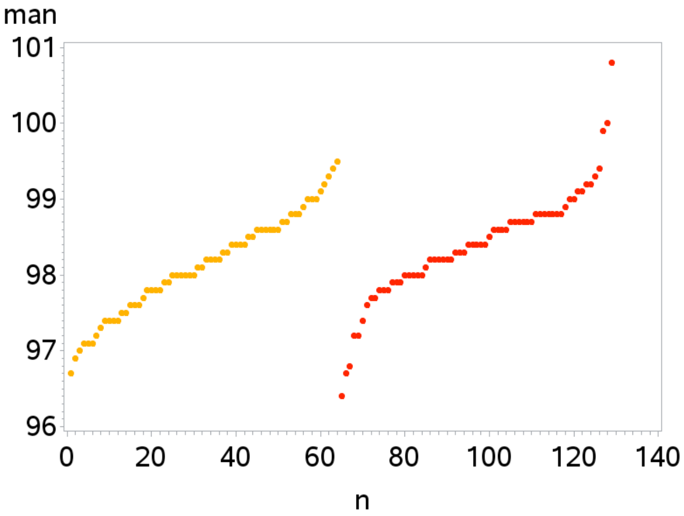
Increasing font size with GPLOT

Increasing font size with GPLOT is easier than SGPLOT, unfortunately, which seems surprising considering that SGPLOT is much more recent. Here it is with the temperature data

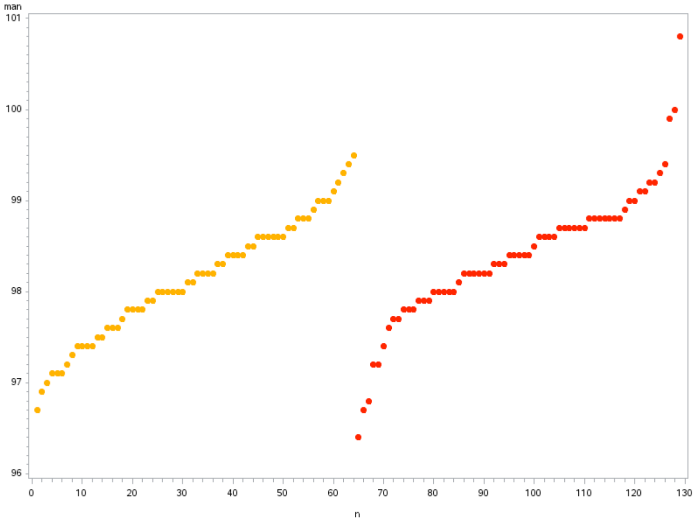
Increasing font size

```
filename foo url "http://math.unm.edu/~james/normtemp.txt";

data new;
  infile foo dlm='09'x firstobs=2;
  input degrees sex $ age;
  n = _n_;
run;

data new2;
  set new new;
  if sex=1 then man=degrees;
  else man=.;
  if sex=2 then woman=degrees;
  else woman=.;
run;

goptions reset=all
ftext='arial'
htext=3.0 /* height of text, determines font sizes */
ftitle='arial/bo'
htitle=1.5
colors=(orange red);

symbol1 value=dot  width=2;
symbol2 value=square  width=2;
proc gplot data=new2 ;
    plot (man woman)*n / overlay;
run;
```

Increasing font size with GPLOT

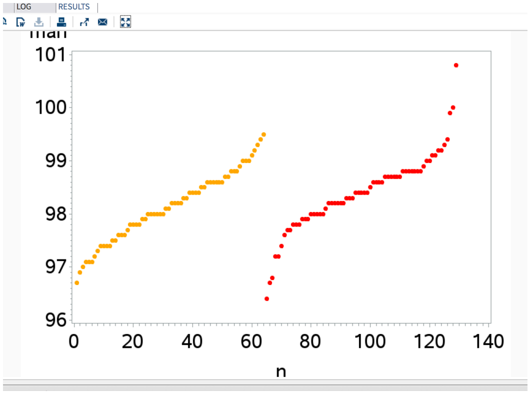


Increasing font size: default with GPLOT



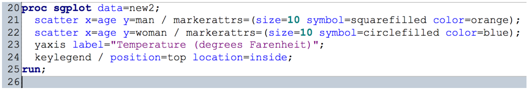
Increasing font size

Unfortunately, the SAS GPLOT output can get cut off in the viewer in SAS Studio. If you copy the image elsewhere (such as to PowerPoint or print the results to a pdf or postscript file), then everything is fine.

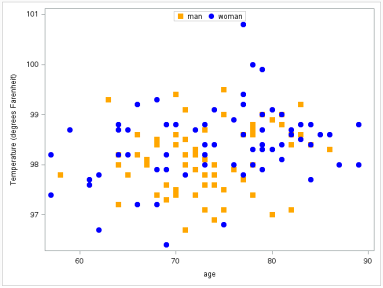


Other ways to visualize Temperature data

We can also plot the data by age. To get the different plot symbols and colors, it is easiest to again have two scatterplots overlaid on each other, which requires having temperatures for men and women in different columns.

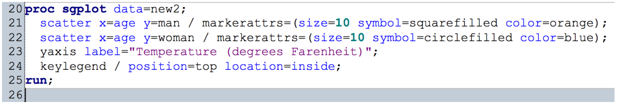


```
proc sgplot data=new2;
  scatter x=age y=man / markerattrs=(size=10 symbol=squarefilled color=orange);
  scatter x=age y=woman / markerattrs=(size=10 symbol=circlefilled color=blue);
  yaxis label="Temperature (degrees Farenheit)";
  keylegend / position=top location=inside;
run;
```

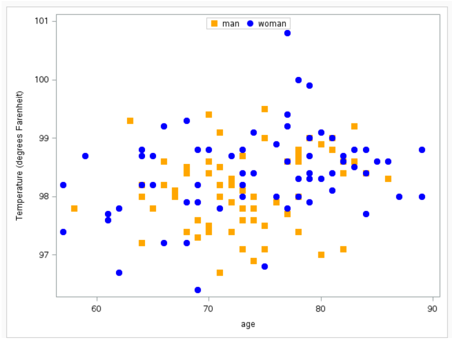


Temperature data

Overlapping histograms are also pretty easy since we have separate columns for men and women.

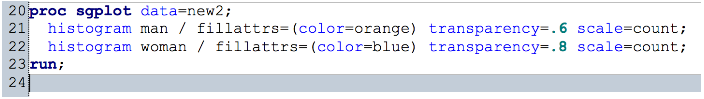


```
proc sgplot data=new2;
  scatter x=age y=man / markerattrs=(size=10 symbol=squarefilled color=orange);
  scatter x=age y=woman / markerattrs=(size=10 symbol=circlefilled color=blue);
  yaxis label="Temperature (degrees Farenheit)";
  keylegend / position=top location=inside;
run;
```

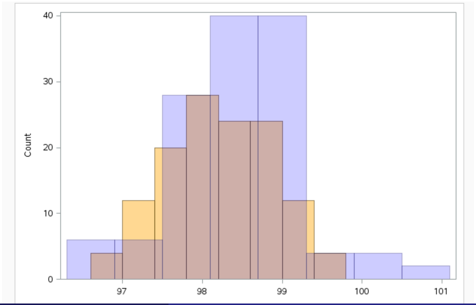


Temperature data

Overlapping histograms are also pretty easy since we have separate columns for men and women. Unfortunately, the `scale=density` option wasn't working, and the areas of the two histograms are not equal.



```
proc sgplot data=new2;
  histogram man / fillattrs=(color=orange) transparency=.6 scale=count;
  histogram woman / fillattrs=(color=blue) transparency=.8 scale=count;
run;

```



Temperature data: ECDFs

Another way of comparing distributions is to compare their estimated CDFs. A very rough way to compare two distributions would be to compare say, their means and variances. Instead of this, we could instead compare their quartiles: the 25th, 50th, and 75 percent largest values. Or you could be more refined and compare deciles: the 10th percentile, 20th percentile, and so on. If two samples come from the same distribution, then these should roughly match.

Instead of determining quantiles based on fixed percentages of the distribution, another approach is to order the data so that if you have n data points, then the percentage of data points less than or equal to the ith data point is $i/n \times 100\%$. We can plot these percentages at each data point, and this is an estimate of the cumulative distribution function.

ECDF

Mathematically, the ECDF can be written as

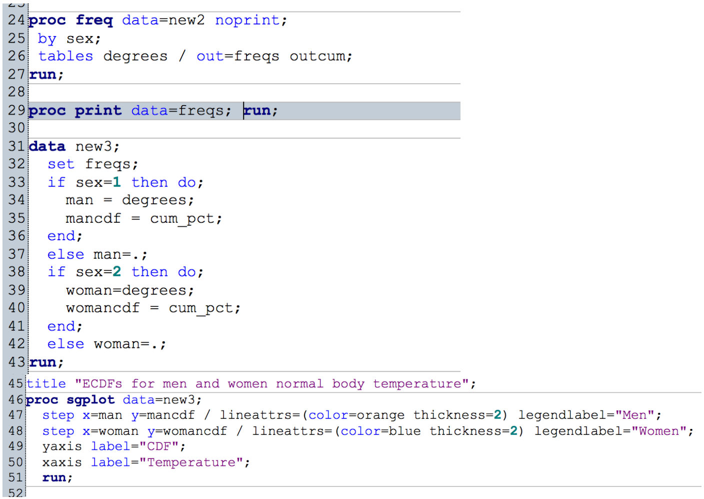
$$\widehat{F}_n(x) = (\text{proportion of observations} \le \text{x}) = \frac{1}{n}\sum_{i=1}^{n} I(x_i \le x)$$

where $I(x_i \le x) = 1$ if $x_i \le x$ and is otherwise 0. The function is plotted as a step function where vertical shifts occur at distinct values observed in the data.

For example, if your data are `1.5, 2.1, 5.2, 6.7`, then $\widehat{F}(3) = \widehat{F}(4) = 0.5$ because 50% of your observations are less than or equal to both 3 and 4. $\widehat{F}(x)$ then jumps to 0.75 at $x = 5.2$.

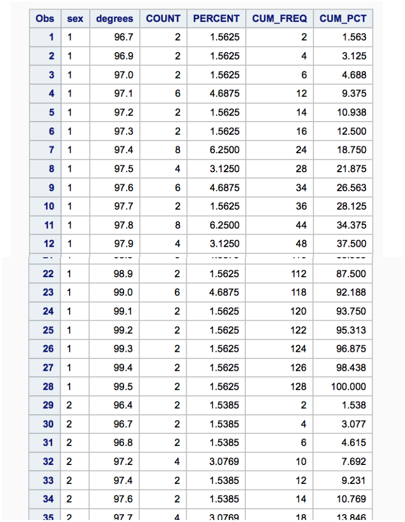
Temperature data

Getting Empirical Compulative Distribution Functions (ECDFs) to



```
proc freq data=new2 noprint;
 by sex;
 tables degrees / out=freqs outcum;
run;

proc print data=freqs; run;

data new3;
  set freqs;
  if sex=1 then do;
    man = degrees;
    mancdf = cum_pct;
  end;
  else man=.;
  if sex=2 then do;
    woman=degrees;
    womancdf = cum_pct;
  end;
  else woman=.;
run;
title "ECDFs for men and women normal body temperature";
proc sgplot data=new3;
  step x=man y=mancdf / lineattrs=(color=orange thickness=2) legendlabel="Men";
  step x=woman y=womancdf / lineattrs=(color=blue thickness=2) legendlabel="Women";
  yaxis label="CDF";
  xaxis label="Temperature";
  run;
```

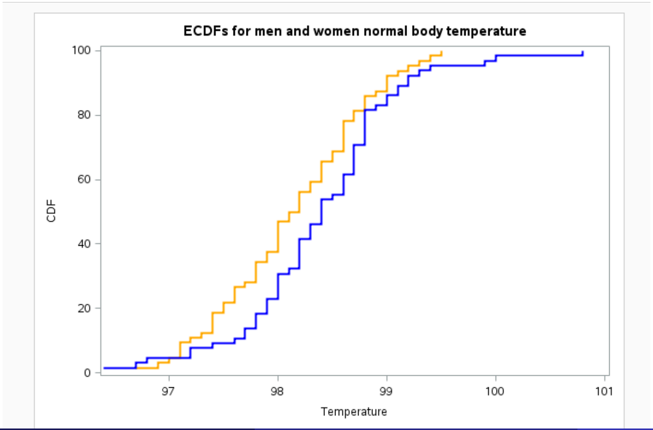
Temperature data



Obs	sex	degrees	COUNT	PERCENT	CUM_FREQ	CUM_PCT
1	1	96.7	2	1.5625	2	1.563
2	1	96.9	2	1.5625	4	3.125
3	1	97.0	2	1.5625	6	4.688
4	1	97.1	6	4.6875	12	9.375
5	1	97.2	2	1.5625	14	10.938
6	1	97.3	2	1.5625	16	12.500
7	1	97.4	8	6.2500	24	18.750
8	1	97.5	4	3.1250	28	21.875
9	1	97.6	6	4.6875	34	26.563
10	1	97.7	2	1.5625	36	28.125
11	1	97.8	8	6.2500	44	34.375
12	1	97.9	4	3.1250	48	37.500
[illegible]	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
22	1	98.9	2	1.5625	112	87.500
23	1	99.0	6	4.6875	118	92.188
24	1	99.1	2	1.5625	120	93.750
25	1	99.2	2	1.5625	122	95.313
26	1	99.3	2	1.5625	124	96.875
27	1	99.4	2	1.5625	126	98.438
28	1	99.5	2	1.5625	128	100.000
29	2	96.4	2	1.5385	2	1.538
30	2	96.7	2	1.5385	4	3.077
31	2	96.8	2	1.5385	6	4.615
32	2	97.2	4	3.0769	10	7.692
33	2	97.4	2	1.5385	12	9.231
34	2	97.6	2	1.5385	14	10.769
35	2	97.7	4	3.0769	18	13.846

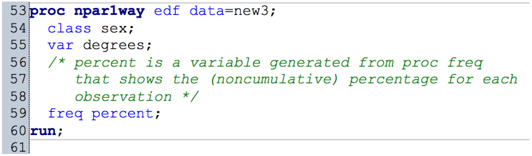
Temperature data

The ECDFs appear to be pretty close. This can be formally tested using the Kolmogorov-Smirnov (K-S) test which looks at the maximum vertical distance between the two ECDFs.

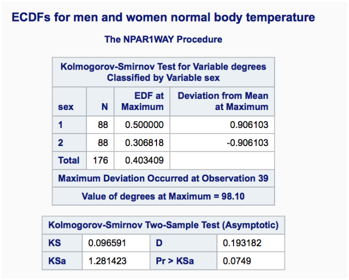


Temperature data

The K-S test gives a p-value of 0.07, so there is weak evidence of a



```
proc npar1way edf data=new3;
  class sex;
  var degrees;
  /* percent is a variable generated from proc freq
     that shows the (noncumulative) percentage for each
     observation */
  freq percent;
run;
```



ECDFs for men and women normal body temperature

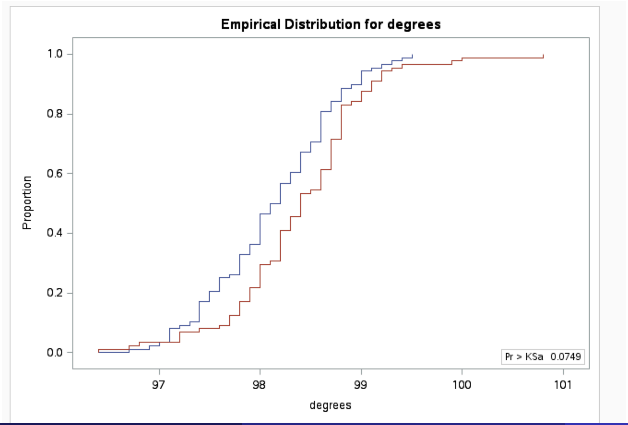
The NPAR1WAY Procedure

Kolmogorov-Smirnov Test for Variable degrees Classified by Variable sex			
sex	N	EDF at Maximum	Deviation from Mean at Maximum
1	88	0.500000	0.906103
2	88	0.306818	-0.906103
Total	176	0.403409	
Maximum Deviation Occurred at Observation 39			
Value of degrees at Maximum = 98.10			

Kolmogorov-Smirnov Two-Sample Test (Asymptotic)			
KS	0.096591	D	0.193182
KSa	1.281423	Pr > KSa	0.0749

Temperature data

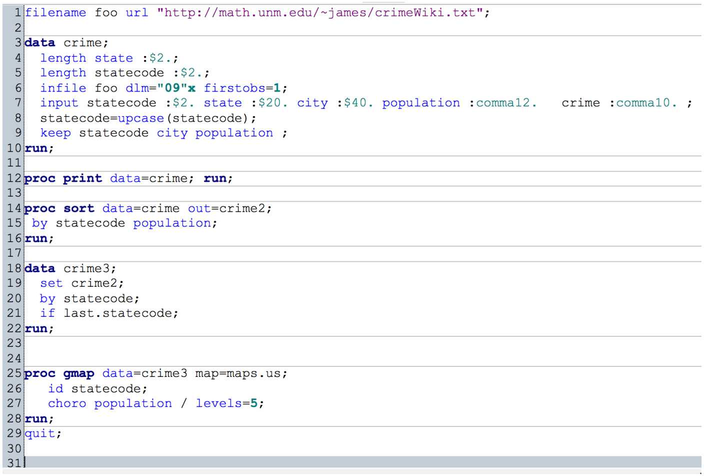
PROC NPAR1WAY also gives a plot for the ECDFs, but it doesn look as nice...The thinner lines and less contrasting colors make the plot more difficult to read.



MAPS in SAS

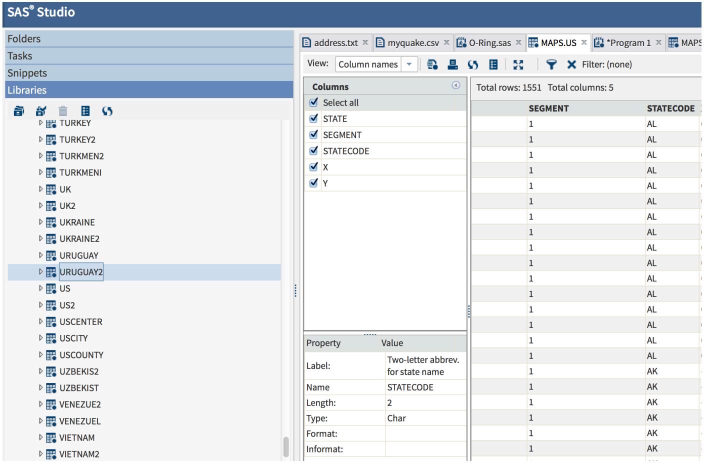
SAS has many built in maps that you can use with statistical information. If you go into the MAPS library, they have maps for the US, by state or with counties, different countries, geographical regions, etc.
The idea is that you use PROC GMAP to read in two datasets: a built in SAS dataset which has geographic regions as a character variable, and your own data that has information on those regions. The ID variable needs to be identical, including the length. For example, if you using state abbreviations like NM, NY, etc., your variable for state ID needs to be exactly 2 characters long, which can be enforced with a LENGTH statement.
Quantitative information for each region can then be displayed using some form of heatmap, for example to represent to population, voter behavior, prevalence of disease, etc.

MAPS in SAS



```
filename foo url "http://math.unm.edu/~james/crimeWiki.txt";

data crime;
  length state :$2.;
  length statecode :$2.;
  infile foo dlm="09"x firstobs=1;
  input statecode :$2. state :$20. city :$40. population :comma12.   crime :comma10. ;
  statecode=upcase(statecode);
  keep statecode city population ;
run;

proc print data=crime; run;

proc sort data=crime out=crime2;
 by statecode population;
run;

data crime3;
  set crime2;
  by statecode;
  if last.statecode;
run;


proc gmap data=crime3 map=maps.us;
    id statecode;
    choro population / levels=5;
run;
quit;


```

MAPS in SAS



SAS® Studio

Folders
Tasks
Snippets
Libraries

TURKEY
TURKEY2
TURKMEN2
TURKMENI
UK
UK2
UKRAINE
UKRAINE2
URUGUAY
URUGUAY2
US
US2
USCENTER
USCITY
USCOUNTY
UZBEKIS2
UZBEKIST
VENEZUE2
VENEZUEL
VIETNAM
VIETNAM2

address.txt | myquake.csv | O-Ring.sas | MAPS.US | *Program 1 | MAPS

View: Column names   Filter: (none)

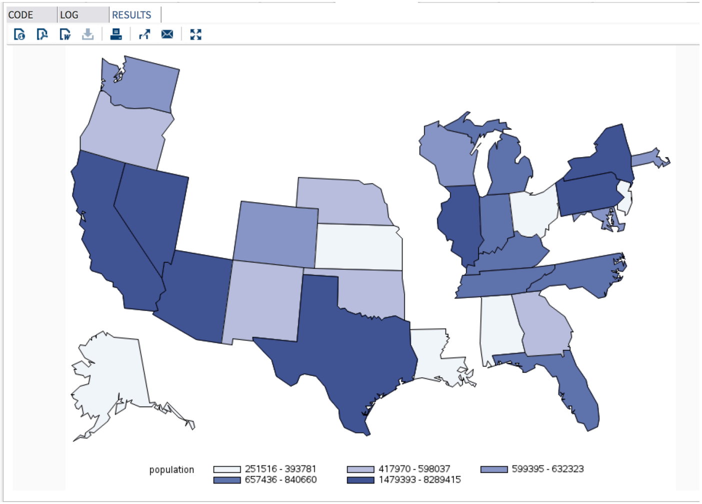
Columns
- ✓ Select all
- ✓ STATE
- ✓ SEGMENT
- ✓ STATECODE
- ✓ X
- ✓ Y

Property	Value
Label:	Two-letter abbrev. for state name
Name	STATECODE
Length:	2
Type:	Char
Format:	
Informat:	

Total rows: 1551 Total columns: 5

SEGMENT	STATECODE
1	AL
1	AL
1	AL
1	AL
1	AL
1	AL
1	AL
1	AL
1	AL
1	AL
1	AL
1	AL
1	AL
1	AL
1	AL
1	AL
1	AK
1	AK
1	AK
1	AK
1	AK
1	AK

MAPS in SAS


CODE
LOG
RESULTS
population
251516 - 393781
417970 - 598037
599395 - 632323
657436 - 840660
1479393 - 8289415

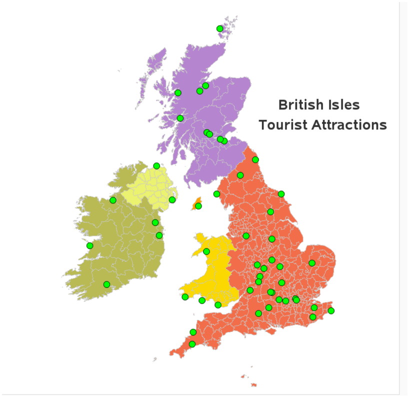
MAPS in SAS

There is a lot more that you can do with maps in SAS. I specified the number of population levels, but then SAS chose the cutoffs. You can also create customized ranges and customized colors. I could have created a different color for states that were missing and maybe gave them a fake population number in the data (like 0 or a negative number) so that they wouldn't be missing on the map.

You can also merge two geographic regions for which there isn't a particular SAS map, such as the US and Mexico.

MAPS in SAS

An example from online...Buttons are links to Google Maps. This was found at http://robslink.com/SAS/Home.htm, and this person has a huge gallery of SAS graphs and plots with source code. This would be a good way of learning more sophisticated things that can be done.


British Isles
Tourist Attractions