36-315: Statistical Graphics and Visualization

Lab 2
Date: January 22, 2002 **Due: start of class January 28, 2002**

1 Introduction

In this lab, you will visualize attributes of your state. In particular, you will look at the distribution of the various Census variables. In addition to making plots, there are five questions.

2 Reading the data

The data at `ciesin.org` is split across multiple `trxxxxa.zip` files. We have automatically collected and concatenated these files, placing the result on the course home page (`www.stat.cmu.edu/~minka/courses/36-315/data/`). If you go there and look in the folder for your state you will find `tracta.zip` and `tractb.zip`. Download and unzip these to get `csv` files. Each file describes the whole state, but on a different set of variables.
For this lab, we will use the variables in `tracta.csv`, specifically `PCTFEMAL`, `PCT40.64`, and `PCTFEMHE`. Use the commands from last time to read the file into S-PLUS and clean up the names. Select one of the variables above by copying it into `x`:

```
x <- frame$PCTFEMAL
```

You'll look at the other variables later.

3 Selecting from Vectors

Some of the elements of `x` may be `NA`, which means undefined. Some functions don't care about this, but others do. You can remove `NA` by using a logical expression, such as

```
x[frame$TOTPOP > 10]
```

or

```
x[!is.na(x)]
```

The result is a new vector. This is another mechanism for selecting from a vector in S-PLUS.

4 Strip plot

A strip plot is a very direct way to represent data; nothing is hidden. To make a basic strip plot, type

```
stripplot(x)
```

A graph window will pop up, obscuring your command window. You can place them side-by-side via

```
Window -> Tile Vertical
```

There will probably be a great deal of overplotting. To spread the data out, add vertical jitter:

```
stripplot(x,jitter=T)
```

Because jittering is random, the vertical position of points will change every time run the command. You can change the vertical range via `ylim`, and the axis label via `xlab`:

```
stripplot(x,jitter=T,ylim=0,ylim=c(0.98,1.02),xlab="Percent female")
```

Note that `ylim` is given twice. This is necessary because of a bug in S-PLUS.

> **Question:** Are there any values which appear to be outliers and/or unusual? Identify them on your plot.

5 Histogram

Histograms allow you to step back from the data and visualize the distribution more abstractly. Histograms represent density with height, rather than texture, which makes certain judgements easier. Histograms generally give a better idea of which points are outliers and the location of density peaks. To make a basic histogram, type

```
hist(x)
```

To use more bins, give the number as a second argument:

```
hist(x,30)
```

To determine an 'honest' number of bins, make a histogram with error bars, by changing `hist` to `bhist`. To use `bhist`, download `www.stat.cmu.edu/~minka/courses/36-315/code/bhist.s` into your work directory and load it via

```
source("bhist.s")
```

Then you can say

```
bhist(x,30)
```

An honest number of bins reveals as much structure as the data can support, but no more. If the error bars are tiny compared to the variation in histogram height, increase the number of bins. If the error bars are large compared to the variation in histogram height, decrease the number of bins. The best number could be 100 or more, depending on how much data you have. You may have to compromise between parts of the distribution that have a different best number of bins.

> **Question:** Identify any bumps in the histogram which suggest outliers. Do they agree with your result for the strip plot?

> **Question:** If the histogram shows multiple modes (density peaks), identify them.

6 Kernel density

It is unlikely that the true distribution is as jagged as the histogram suggests. The jumps in height are also distracting, from a presentational perspective. An alternative to the histogram is a kernel density: a summation of kernel functions placed on the observations. You can compute a kernel density estimate via `density`, and plot it via `plot`:

```
plot(density(x),type="l")
```

To see what `type="l"` does, try plotting without it. The change the width of the kernel function, which is analogous to choosing the number of histogram bins, give a `width` argument to `density`:

```
plot(density(x,width=2),type="l")
```

Choose the width so that the kernel density shows approximately the same details as the best histogram above. Note that width is measured on the same scale as `x`. To get in right ballpark, you can call the function `bandwidth.bcv` which will return a width:

```
bandwidth.bcv(x)
```

`density` has various options for setting the width and kernels. You can get documentation on this function via

```
?density
```

> **Question:** Identify any bumps in the density which suggest outliers, and if the density suggests multiple modes, identify them.

7 Other variables

Repeat the above and make a strip plot, histogram with error bars, and kernel density for all of the variables: `PCTFEMAL`, `PCT40.64`, and `PCTFEMHE`.
Print and submit your plots. There should be $3 \times 3 = 9$ total. Make sure that the axes are labeled correctly using `xlab`. Click on the plot window and select `File -> Print` or `File -> Export Graph...`.

> **Question:** Which variable varies the most across the state (has flattest density)? Which varies the least (has sharpest density)?

8 Notebook

At this point, you may want to start keeping a personal notebook of anomalous features in your state, or interesting questions to ask about your state. It will come in handy as a source of ideas for the final project.