Mathematics 375 – Probability Theory
Computer Lab Day 1 – Introducing R
September 5, 2011

Goals

The goals of today's lab are:

- to introduce some basic features of the R statistics package that we will be using some this semester and much more heavily next semester, and
- to introduce some additional descriptive statistics and graphics for understanding the "shape" or "distribution" of a data set.

Background on R – Getting Started

The instructions here are geared toward the Windows version of R running in the Haberlin 136 lab. R is also installed on the Math/CS Linux network (Swords 219 lab, other machines on Swords 3). Since R is a free, open source package, you can also obtain versions to run on your own computers if you want, and there are Windows, Mac-OS, and Linux versions available. These have the same functionality but slightly different interfaces.

In HA 136, when you double-click the R shortcut from the standard desktop, you will launch the R-GUI window. Inside this is a workspace window with a white background. This is *not* entirely like a Maple worksheet, unfortunately. It *only* allows you to

- enter commands, and
- view numerical output.

It does not let you create integrated documents, and graphics are displayed in separate windows within the R-GUI, as you will see.

You can *print* the workspace window or any graphics window at any time. Just highlight that window and press the toolbar printer button.

To save your work, with comments and answers to questions (e.g. for the lab reports you submit for the lab day assignments), I recommend

- opening another editor window (either WordPad or NotePad are fine for this on Windows systems), and
- using it to create a text file that will be your permanent record of your work.
- When you have a result you want to save from the R workspace, copy both the input command and the output and paste into the text file.
- Any time you want to add a comment to explain a calculation or answer a question, enter that directly into the text file.

- Save that text file on your personal network P : drive and print it when you want a hardcopy (from WordPad or NotePad).

A First R Session

In the lab today, you will basically work through some examples illustrating features of R and some of the descriptive statistics we have talked about so far in the course. You will keep a record of what you do and submit it as the first lab assignment.

Lab Exercises

1. In its most basic form, R provides a command-driven "statistical calculator" in which you type in data and commands to generate numerical and graphical output that are displayed. When you have the R window open, note the > input prompt after the opening message. This is where you should start. This exercise works through problem 1.35 in our text (Wackerly, Mendenhall and Scheaffer). First we need to get the systolic blood pressure readings into R so that we can work with them. The basic way is to enter (after the > prompt):

```
bpdata <- c(172,140,123,130,115,148,108,129,137,161,123,152,133,128,142)
```

   and press ENTER. Note: the $<-$ is formed by typing the $<$ and $-$ characters right next to each other. It is the *assignment operator* in R. This takes the data on the right and assigns it to the name on the left. The c, followed by the list of numbers, separated by commas is R's way of constructing *an ordered list or vector*. You should just get another input prompt if everything went OK with this (error messages if not). If you did get an error, try again, and try to follow the above exactly. You can verify that the numbers are entered correctly by typing in the name `bpdata` as a command; the numbers will be displayed in a row.

   a. Part a of the problem says to estimate the SD of the data using the range (max minus min). The empirical rule we discussed says that a *very rough* estimate for the SD is

$$SD \doteq \frac{1}{4}(range).$$

      (This tends to work better for *small* data sets without "outliers.") In R we can do this computation very easily using built-in functions: Type in

```
(max(bpdata) - min(bpdata))/4
```

      and press ENTER to see the result.

   b. Part b of the problem says to compute the actual mean $\overline{y}$ and s of the data. This can also be done in R (in several different ways!): Type in

```
ybar <- sum(bpdata)/length(bpdata)
mean(bpdata)
s <- sqrt(sum((bpdata - ybar)^2)/(length(bpdata) - 1))
sd(bpdata)
```

and press ENTER after each line to see the results. Note the way the sum function works in a very nice way here – in the computation of `ybar`, we only need to put in the name of the data list; the sum of all the elements is computed by default. Then in the computation of `s`, we can subtract *each entry* of the `bpdata` vector from `ybar` as shown! Of course we will usually want now to use the built-in functions `mean` and `sd`, but the first of each pair of computations shows how we can also use more primitive functions in R when we need them.

c. Tchebysheff's theorem is a result about the distribution of data sets that complements the "empirical rule." It says in a particular case that *there will always be at least* 75% *of the numbers in a data set within 2 standard deviations of the mean.* We will prove this and more general statements later in the semester. For now, let's use R to check that this is true here. In its most basic use, R can function as a simple numerical calculator(!)

```
upperb <- ybar + 2*s
lowerb <- ybar - 2*s
```

How many of the data values are in the range $[\overline{y} - 2s, \overline{y} + 2s]$, though? We could just count by hand, of course, with a data set this small. But R also contains a very rich collection of facilities for selecting, subsetting, and massaging data in numerous ways. Here is a basic way to select the portion of the data set in the range we want:

```
middle <- subset(bpdata, bpdata <= upperb & bpdata >= lowerb)
length(middle)/length(bpdata)
```

(This should be pretty self-explanatory; ask if you don't understand what this did!) Did Tchebysheff's theorem "work" here?

2. Now let's look at a different data set. Problem 1.36 in our text deals with the number of parasites found in each of a sample of 100 individual foxes. The information is given there in table format:

Parasites	0	1	2	3	4	5	6	7	8
Foxes	69	17	6	3	1	2	1	0	1

So we want to create a data set containing 69 0 entries, 17 1 entries, etc. Clearly this is going to be tedious and error prone if we do it the most basic way(!) Fortunately, R lets us construct lists in more flexible ways besides just listing the entries. We can indicate how many times to repeat a given value if we want, like this:

```
foxpar <- rep(c(0,1,2,3,4,5,6,7,8),c(69,17,6,3,1,2,1,0,1))
```

Of course `rep` stands for "repetition" here. Note that the first list inside the `rep` gives the values of the number of parasites – the first row in the table format – and the second list gives number of times each value appears in the data – the second row. The first list could also be abbreviated as `0:8` (R's notation for a list of successive integer values).

a. Part a of the problem says to construct a relative frequency histogram for the number of parasites per fox. To generate this in R, let's begin by entering

```
hist(foxpar)
```

The histogram will be generated in a separate graphics window. This is a "default" version of the command. The *hist* command takes a number of options that control how the "bins" are selected and how the histogram is drawn as well. You will often want to modify the default results. For instance, looking at the output, how were the bins chosen here? Is that optimal for showing the different numbers of foxes with each number of parasites? Suppose we want bins centered on the integer values $0, 1, 2, 3, 4, 5, 6, 7, 8$ as we did in another example in class. We can put in an option to make R do it that way too. The `breaks` option takes a list of values and makes those the boundaries of the bins for the histogram:

```
hist(foxpar,breaks=c(-0.5,0.5,1.5,2.5,3.5,4.5,5.5,6.5,7.5,8.5))
```

How is this different and why is it better than the default way that R drew the histogram? Include a printout of this histogram in your lab report. (See above for printing directions.)

b. To practice using the commands we talked about in problem 1, calculate $\overline{y}$ and s for this data set. How many of the numbers were within two standard deviations of the mean in this case?

c. The list of breaks for the histogram can be anything in R. What happens if you change the command above to

```
hist(foxpar,breaks=c(-0.5,0.5,1.5,2.5,3.5,4.5,5.5,8.5))
```

(deleting the 6.5 and 7.5 break points, so we're lumping together the counts for the values 6,7,8 parasites)? What is different about the new histogram that R drew in this case? Look at the graph carefully: do the heights of the boxes still represent *frequencies*? (Hint: what they do represent are *densities* that are computed by taking the frequency count and dividing by the width of the box in each case. What is true if we add the areas of the boxes drawn this way? We will see other reasons why this is a preferable

way to draw histograms with unequal bin sizes later.) If we *really want frequencies*, we can make R do it that way:

```
hist(foxpar,breaks=c(-0.5,0.5,1.5,2.5,3.5,4.5,5.5,8.5),freq=TRUE)
```

However note that now R generates a warning message saying the AREAS in the plot are "wrong." Think about this. Why is the last box in this version of the histogram *misleading* if drawn this way?

3. In almost all real-world analysis of data, the data entry and analysis steps are separate. So, for instance, someone might enter the data into a computer file in some specified format. Then a statistician would read that data into a program like R to analyze the data and look for patterns or try to draw inferences about the situation described by those measurements. We will see an example of that in this question. Enter this command to read in a modified version of the data set about lifetimes of radio transmitter-receivers from problem 1.25 in the text.

```
lifetimes <- read.table("http://mathcs.holycross.edu/~little/
      ProbStat1112/R/Lab1Data.dat",header=TRUE,sep=",")
                    attach(lifetimes)
```

(Note: the first two lines here are part of one command, just keep typing until the final close paren.)

The modification is that the numbers have been explicitly grouped into 8 columns of 11 numbers each, with a column heading that is a character string. The `attach` command above sets up the headings as names for the different columns. For the purposes of this question, we will think of them as 8 separate samples of lifetimes, called `s1, ..., s8`. Each of those 8 data sets will allow for the same operations as the data sets from problems 1 and 2 above.

   a. In addition to the mean and standard deviation, statisticians use the median and other percentiles to understand how data sets are distributed. R has multiple ways to compute these, including a default `summary` command that computes the min, max, median, mean, 25th and 75th percentiles all together:

```
summary(s2)
summary(s8)
```

   Note that the 25th percentile is called the 1st quartile and the 75th percentile is called the 3rd quartile in R. What is your first impression about the distributions of these two samples, given these summary statistics?

   b. Plot the histograms for each of `s2, s8` to see if your impression is confirmed.

c. There are other very useful graphical ways to represent data besides the frequency or density histogram. For example, statisticians may use what are called "box and whisker" plots for this purpose. Try

```
boxplot(s2)
```

to see one of these plots. You can figure out what the plot represents if you carefully look at the vertical scale together with the values returned by the `summary` command for this data set. What is the "box;" what are the "whiskers?" Can you guess what the circles represent?

d. Often we want to compare the distributions of two or more data sets. We can do "parallel box and whisker plots" in R like this:

```
par(mfrow=c(1,2))
   boxplot(s2)
   boxplot(s8)
par(mfrow=c(1,1))
```

(Note: The `par` commands set up the graphing so that the box plots come out side-by-side, then return the graphics format to the original state for any graphing that comes later.) Hand in a printout of this plot with your lab report.

e. Describe in words your conclusions about the distributions of these two data sets. (Comment: There are apparent differences and your job is to try to describe them. Whether those are *significant enough* to justify a conclusion about a possible difference between the two samples is the kind of question the statistical techniques we will learn next semester are designed to help answer.)

Assignment

Lab reports containing all input, output, and answers to the questions posed above are due in class no later than Monday, September 12. If you do not finish during the hour today, you can return to HA 136 any time it is not in use by another class.