Homework assignment # 2

1) Your first real R assignment (*see R comments/instructions at the end*):

a) Download/save the data set from the course web page (look for the link next to homework 2). These are real data on the length of snapping turtles.

http://mason.gmu.edu/~alaemmer/bio214/main.html

b) Read the data into R.

c) Calculate the mean and standard deviation for the length of our turtles (both sexes combined - see also part (e)).

d) How many males are there? Females?

e) Suppose we want to know the means for females vs. males. Go ahead and calculate the mean for females and for males.

f) Repeat (e), but now do the median, standard deviation, and variance for males and females.

g) Generate a histogram of this data set.

h) Finally, make a parallel boxplot for these data (males vs. females).

2) Here are some (simulated) data on the length (in cm) of 16 blue ringed octopuses (*Hapalochlaena lunulata*). Construct a boxplot of these data. ***Do not use R****:*

17.9 17.5 12.8 14.6 16.3 18.0 17.6 16.0 16.6 15.6 15.4 17.6 16.4 16.0 20.0 16.9

3) Here are the heights (in meters) of sycamore trees and pin oak trees. Construct a correct parallel boxplot for these data. ***Do not use R****:*

sycamores: 31.7 13.7 21.0 29.6 32.9 7.3 30.8 43.3 29.9 15.8 14.6 23.2 22.6

Pin oaks: 25.0 26.2 21.3 50.2 25.0 23.8 25.9 28.3 19.5 25.6 10.1 18.6 23.2 19.5

4) Pick random numbers between 0 and 9 inclusive (i.e., 0 and 9 should be part of the numbers you pick). Do this twice:

a) Ask R to give you 100 random numbers (*see instructions below*).

b) Now ask R to give you 1,000,000 random numbers.

c) Now make ***two*** bar plots, one for (a) and one for (b):

You want to plot the numbers 0 to 9 on the x axis and the number of times you got each number on the y axis.

d) Which barplot shows data that look more random? ***Why?***

*(Hint: if the data are truly random would you expect each digit to show up about the same number of times? Why or why not? **Make sure you understand this!**)*

e) Explain what happened as you went from (a) to (b).

5) Here are some (simulated) data on the maximum age distribution in rabbits:

age:	1	2	3	4	5	6	7	8	9
frequency:	234	423	378	235	97	58	21	6	0

Give the following (watch out for $<$ vs. $\leq$):

a) $\Pr(Y > 6)$ **b)** $\Pr(3 < Y \leq 6)$ **c)** $\Pr(Y \leq 3)$ **d)** $\Pr(Y \leq 8)$ **e)** $\Pr(Y < 8)$

f) $\Pr(Y = 7)$ **g)** $\Pr(Y > 5)$ **h)** $\Pr(Y = 9)$

i) Add (a), (b) and (c). Are you surprised? *Why or why not?*

6) You study for an exam, but run out of time. Still, you manged to learn 70% of the material. The exam is all multiple choice, with each question having 5 choices.

a) If you come across a question for which you don't know the answer, what's the probability of getting the answer right?

b) If you come across a question for which you do know the answer, what's the probability of getting it right (hint: don't “over think” this - what does it mean if you **know** the answer...)?

c) What is the overall probability of getting ***any*** question correct?

7) In some parts of Africa, female giraffes (*Giraffa camelopardalis*) make up about 62% of the population. Suppose a genetic trait is more common in females and occurs about 71% of the time. In males this trait occurs 13% of the time.

(a) What is the probability that a randomly chosen giraffe will have the trait?

(b) Suppose a randomly picked giraffe has the trait (i.e., you pick a giraffe without paying attention to the sex), what is the probability this giraffe is a female?

Problems are due in recitation the Monday, June 10th.

Remember: Copying answers from other students, web sites, or from previous classes is a violation of the honor code

R comments/instructions:

Be a bit patient and read the instructions carefully. This isn't difficult if you're not in a rush.

Reading data into R:

Preliminary comment (you don't need to worry about this right now): the turtle data that are posted are already in the correct format for R. If you're trying to get data from Excel (or another spreadsheet) into R, then you may need to go through an extra step. You might need to export the data in a comma delimited file - if this sounds like gibberish, please look at chapter 2 in the text.

Using the RStudio menu:

Comment: the nice thing about RStudio is that you can use it just like regular R; if you follow the instructions below and put them in the script window, and then run them, they will work!

Click on "Import Dataset" on the top right window.

Select "From Text File".

This should open a familiar file open dialog.

Find where you saved the file and click on it. If you can't find your file, see below under "*Where is your file*" for some advice.

You should now see a window that says "Import Dataset" at the top.

On the right side of this you'll see two smaller windows. The top shows what the file actually looks like on your hard drive, the bottom shows what it will look like when you import it.

On the left side you see some options. Make sure Heading is set to "Yes", Separator is set to "Comma", Decimal is set to "Period", and Quote is set to "Double quote".

Click on Import, and your data should be imported into R. Your dataset should be called something like "snapping"

Incidentally, note that RStudio will put the command used to read the data into the command window (it'll look a lot like the command given in the instructions below).

If you want to do this from the command line without using the R-Studio menu:

Read the data into R using the "read.table" command (please don't just type this as given; read the instructions first and remember that R is case sensitive!):

```
snapping <- read.table("path-to-file/file-name", header = TRUE, sep = ",")
```

So what does it all mean?

read.table is the command that R uses to read data that has more than one column (our data has three columns) "path-to-file/file-name" is the location and name of your data. This is the bit that will probably give you the most headaches (no matter your OS, note that R uses / not \ in its path).

Where is your file?

In Windows, it's most likely on the desktop or in your "my documents" folder. If you saved the file to your desktop, you'll need something similar to the following:

"C:/Documents and Settings/*your-user-name*/Desktop/snapping.csv"

(all on one line)

In Mac/OS, it's most likely in your User directory. Something similar to the following should work:

"/Users/*yourname*/snapping.csv"

In Linux, you probably know where it is (it's most likely somewhere in your home directory, perhaps in the Desktop, Documents, or Download folders if you have these). You'll probably use something like:

"home/*your-user-name*/snapping.csv"

or possibly:

"home/*your-user-name*/Download/snapping.csv"

All OS's: If you have trouble, you'll need to open your file browser and look around for it until you find it. When you find it, take careful note of the location (usually visible near the top of the file browser window somewhere).

header = TRUE tells R that the names of the variable are in the first row (in our case, "sex", "length", and "body_temp"). Not all data sets will have variable names.

sep = "," tells R what character is used to separate data on the same line. If you look closely at our data set, you'll see that all the values are separated by commas.

Finally notice that the bit up front, "snapping <-", tells R to read the data into a variable called "snapping"

If you don't do this, R will read the data but then immediately forget it.

Calculating means and standard deviations:

Once your data is in R, this is fairly simple. The only difficult part is knowing what your variables are called.

If the name of your data set is "snapping", then you can do:

```
mean(snapping$length)
```

"snapping" tells R to look inside the "snapping" dataset, and $length tells R which variable you want the mean for. The “$” symbol tells R to look for “length” in the dataset “snapping”.

To get the standard deviation, use "sd" instead of "mean". Otherwise the format of the command is the same. For the median, use "median", and for the variance use "var".

(The "var" command actually works a bit differently than the others, but it should work as given for this example - but you might be surprised if you use it elsewhere).

Counting males vs. females:

For this you can use the "table" command. This “tabulates” categorical data (tells you how many things you have for each category). “sex” is a categorical variable, so you can try something like:

```
table(snapping$sex)
```

You'll get more examples using the table command later in the semester.

Calculating means (and standard deviations) separately for females and males:

To do this, you need to use the tapply command as follows:

```
tapply(snapping$length, snapping$sex, mean)
```

Notice there are three parts in the ( ).

The first, snapping$length, is the variable you want the means for.

The second, snapping$sex, is the variable for which you want separate means (females & males). Note that this variable is categorical.

The last part (mean) is what you want R to calculate.

In other words, R will use sex to separate out the lengths for males and females, and then calculate the mean.

To get the standard deviations, just use sd instead of mean in the last part of tapply.

Histograms with titles

Now let's do a histogram. The command is hist. So you'd do:

```
hist(snapping$length)
```

However, let's make it look a little better. Here's another version of the hist command. Run it to see what it does, then go ahead and substitute something you like for “a”, “b”, “c” and “yellow” below, and run it again (*don't hand in a histogram with a, b, c and yellow - **make it look nice!***).

```
hist(snapping$length, ylab = "a", xlab = "b", main = "c", col = "yellow")
```

Boxplots

To make boxplots, you use the boxplot command:

```
boxplot(snapping$length)
```

This gives you a boxplot for all turtle lengths. But the problem above asks you to make a parallel boxplot of males vs. females:

```
boxplot(snapping$length ~ snapping$sex)
```

If you want the boxplot to be horizontal you can add the horizontal option (this also works with a regular (non-parallel) boxplot:

```
boxplot(snapping$length ~ snapping$sex, horizontal = TRUE)
```

You can use the same options as for histograms (ylab, xlab, main, col) if you want to label your plot - but be careful, if you make your boxplot horizontal ylab and xlab are *always* the vertical and horizontal axes respectively.

Random numbers in R:

To generate random numbers in R, you can follow the instructions in the notes or modify the following:

Generate a list of numbers between 0 and 9 (inclusive):

```
y <- seq(0,9)
```

(If you want to look at y, you can just type "y" in R - it's just the digits 0 through 9)

Now suppose you want to take a sample of size 75 from this sequence:

```
y1 <- sample(y,75, replace = TRUE)
```

The replace = TRUE part tells R that each time you sample a number, you put it back in your sample (otherwise you only have 10 numbers to sample from!!)

Finally, you can use R to count how many times you got each number:

```
table(y1)
```

That should help you construct your barplot.

Barplots in R:

Making barplots of our random numbers is not quite as easy as using the hist command, but it's not that difficult.

Suppose you have 1,000 digits from 0 to 1 (inclusive) in the variable y1. The you would do:

```
barplot(table(y1), names.arg = seq(0,9), col = "gray", ylab = "Frequency", xlab =
"Digit")
```

"barplot" is obviously the name of the command.

The first thing you give it is the numbers of each of the things you're interested in (that's what "table" is doing - it's giving the barplot command the number of 0,'s, 1's, 2's, etc.)

The second thing is the "names.arg"; this gives the names you want for each of the bars on the x-axis. Notice it has the digits 0 through 9, and this is what you'll find your bars labeled with.

The rest of the barplot you should be able to figure out yourself, since you've used similar arguments for the histogram command.

(Comment: There are ways to make each bar a different color and so on, but we don't really need that right now).