RESEARCH SCHOOL OF FINANCE, ACTUARIAL STUDIES AND APPLIED STATISTICS

INTRODUCTORY *R* WORKSHEET 3

Worksheets 3 and 4 are more advanced and are designed for those who wish to learn more about R. It is not required that you work through these two worksheets, as Worksheets 1 and 2 should be sufficient to allow you to get started in your course. All the R relevant to your course will be covered during semester. But, if you want to learn more or practice more, read on.

1. *Randomness.* Develop R commands to do the following:

(a) Generate 1000 random values from a chi-squared distribution with 18 degrees of freedom using the `rchisq()` function, and put the result into a vector called `chivals`.

(b) Construct a histogram from the random vector `chivals`.

(c) What are the smallest and largest values in `chivals`?

(d) The histogram command can be forced to use certain break points rather than the default breakpoints which *R* chooses. For example, the command

```
hist(x,breaks=seq(floor(min(x)),floor(max(x)+1),1))
```

will create a histogram of data values in some given vector `x`, but it will use bins of width one, spanning the range between the smallest and largest integers bracketing the range of `x`. By adjusting the bin width (that is, the third argument in the `seq()` function above), you can achieve a smoother or rougher looking histogram than that produced by the default binwidth. Experiment with various binwidths by varying the line given above and see what binwidth yields the "smoothest-looking" histograms. What do you notice about the histograms as the binwidth increases? decreases?

(e) Create a new vector which contains the average of 25 random vectors of length 1000 from a chi-squared distribution with 18 degrees of freedom (i.e. using 24 more random vectors in addition to your original one from part (a)) by using the `for()` command to loop as follows:

```
for(i in 1:24) {
chivals <- chivals+rchisq(1000,df=18) }
```

Note that we can overwrite an object with itself as we calculate to save space and names.

Now, create the averages by dividing `chivals` by 25 and placing the result in a vector called `chiavg`. Examine a histogram of these averages using a binwidth of length 0.5. What should the mean of these values be? What about their standard deviation? (Recall that the mean of a chi-squared random variable is equal to its degrees of freedom and the variance of a chi-squared random variable is equal to twice its degrees of freedom.)

(f) According to the Central Limit Theorem, the histogram above should have a normal distribution with the mean and standard deviation you calculated above. To see whether this appears to be true, we will plot a picture of the appropriate normal density curve onto our histogram. To start, take a vector that spans the range of the values in `chiavg`, for example, the vector

```
grid <- seq(floor(min(chiavg)),floor(max(chiavg)+1),0.1)
```

is such a vector. Find the values of the appropriate normal density corresponding to the values in `grid` using the `dnorm()` function and store these in a vector called `normals`. Now, use the `lines()` function to superimpose the plot of `grid` versus `normals` on the histogram constructed in (f). What do you notice? If you said "not much", you are right, since the normal density we plotted has a completely different scale to the histogram. This is because the histogram uses raw frequencies rather than relative frequencies. To account for this, we will want to "blow up" the normal density we plotted by a factor of 1000 (the total number of simulated values) multiplied by the binwidth (in this case 0.1). So, now superimpose on your

histogram the plot of `grid` versus `100` $\times$ `normals`. You may want to compare a plotted density against histograms of `chiavg` using different binwidths to see which corresponds most closely to the true curve. Do you believe the average values in `chiavg` are normally distributed?

(g) Let's test the last idea that the averages in `chiavg` are credibly normal. For this task, we want to make a *normal quantile* or *q-q plot*. The idea of a *q-q plot* is basically to match up the ordered observed values with what we would *expect* to be the ordered values from a normal distribution with the appropriate mean and variance. These latter values are called *normal scores*. Construct a *q-q plot* of the data in `chiavg` using the function `qqnorm()`. What should the *q-q plot* look like if the data do indeed follow a normal distribution? Do you think the simulated data might reasonably have come from a normal distribution?

2. *Functions.* In this part of the worksheet, we are going to write our own function. This function is going to investigate how the mean and variance of a sample of numbers is influenced by the exclusion of several user-chosen data points. As such, we shall name the function `influence()`. Such a function will be useful for identifying potential outliers and influential points in a data set.

In general, functions have the form

```
function(argument1, argument2, ...){
    <text of function goes here>
}
```

The value of the last line of the contents of the function will be its returned value. As a simple example, recall the following function which finds the square of its argument:

```
square <- function(x){
    x^2 }
```

The last (and only) line of the function is the value returned by the function. Alternatively, we could have written this function as:

```
square <- function(x){
    y <- x*x
    y }
```

Since the first line of this function is an assignment and therefore has no value, we must include the second line so that the function will return the desired result.

To start our function `influence()`, we must first think of what our arguments need to be. The function must take as input the original data to calculate the mean and variance and also a vector of numbers indicating which points to exclude. Now we are ready to begin the function

```
influence <- function(samp,excl){
```

Now, our first step will be to check that there are no repeats in the `excl` argument as there is no need to exclude a point more than once. So the first line of our function will be:

```
exc <- unique(excl)
```

The `unique()` function takes a vector input and returns a vector with any repeat values removed.

We now need to check whether any values in the `exc` vector are invalid, i.e. whether they are below one or larger than the number of data points in the given `samp` vector. If the values are invalid, then we need to print a warning message. To do this, we will use the `if()`, `else if()` and `else()` functions as follows:

```
if(min(exc)<1) {
    print("Invalid Point to be Excluded - Index too small") }
else if(max(exc)>length(samp)) {
```

```
    print("Invalid Point to be Excluded - Index too large") }
else {
```

Now, once we have established that the `exc` vector is valid, we can proceed to calculate the mean and variance for the the entire data set

```
mn <- mean(samp)
vr <- var(samp)
```

and the mean and variance using the reduced data set

```
mn.red <- mean(samp[-exc])
vr.red <- var(samp[-exc])
```

Notice that we make use of an important aspect of the "square bracket" notation for vectors; namely, that if negative values are given, then *R* returns a vector with the specified elements removed. In other words, while `x[1]` indicates the first element of the vector `x`, t h e notation `x[-1]` indicates a vector which contains all of the elements of the vector `x` except for the first. Of course, the above description generalizes to the case where the value in the square brackets is a vector of numbers, so that `x[c(1,2)]` indicates the first two elements of the vector, while `x[-c(1,2)]` or `x[c(-1,-2)]` indicates all of the elements of the vector except for the first two.

Lastly, we must calculate the difference in the slopes and intercepts and return this value:

```
c(mn,vr) - c(mn.red,vr.red) } }
```

So, the entire function would look as follows:

```
influence <- function(samp,excl) {
  exc <- unique(excl)
  if(min(exc)<1) {
    print("Invalid Point to be Excluded - Index too small") }
  else if(max(exc)>length(samp)) {
    print("Invalid Point to be Excluded - Index too large") }
  else {
    mn <- mean(samp)
    vr <- var(samp)
    mn.red <- mean(samp[-exc])
    vr.red <- var(samp[-exc])
    c(mn,vr) - c(mn.red,vr.red) } }
```

Recall the data set *worksheet2.women*. Use this data to try out the new function `influence()`. Can you find any points (singly or in groups) which have a large effect on the values of the mean and variance of either of the two measurements (i.e., height or weight)?