**DS 710**

**Homework 10**

**R assignment**

1. In this problem, you will work with the “cleaned” version of the US News and World Report data on colleges and universities, which you created in Homework 9.
2. Read the data into R and attach it.  Use *length* and *which* to determine how many schools have a per-student instructional expenditure higher than their out-of-state tuition.  Then use control flow to answer the same question.  (Check that the two methods give the same answer!)

**Using length and which:**

> length\_per\_student\_higher\_tuition <-length(which(usNews$Instructional.expenditure.per.student>usNews$Out.of.state.tuition))

> length\_per\_student\_higher\_tuition

[1] 457

**With control flow:**

> counter=0

> for (i in 1:nrow(usNews)){

+ if(!is.na(usNews[i,34])&!is.na(usNews[i,23])){

+ if(usNews[i,34]>usNews[i,23]){

+ counter<-counter+1

+ }

+ }

+ }

> counter

[1] 457

1. Use *system.time* to compare the running times of the two methods you wrote in part a.  Iterate each method enough times that you can see a difference in the running times.  Report the user time + system time for each method.  Which is more efficient?

> system.time(for(i in 1:10000000)length\_per\_student\_higher\_tuition)

user system elapsed

0.186 0.017 0.203

> system.time(for(i in 1:10000000)counter)

user system elapsed

0.179 0.009 0.187

**counter is more efficient**

1. Consider three different methods of finding the mean of each numeric column of the data:
2. Using apply() and the built-in function mean()

average=usNews[,c(4:35,37:38)]

> average=average[!is.na(average)]

> average<-apply(usNews[ ,c(4:35,37)],2,mean,na.rm=T)

> average

1. Using apply() and a function you write, called mymean(), which takes the sum of all of the non-missing values and divides by the number of non-missing values, without using the built-in function mean().

mymean<-function(x){

+ x<-x[!is.na(x)]

+ average=sum(x)/length(x)

+ return(average)

+ }

> average2<-apply(usNews[,c(4:35,37:38)],2,mymean)

> average2

1. Using a for() loop to iterate over the numeric columns, and a for() loop inside it to iterate over the values within that column, without using the built-in functions mean() or sum().

> mymean2<-function(usNews){

+ usNews\_remove\_non\_numeric <-usNews[,-c((1:3),36)] #removes non-numeric columns

+ colAverages = numeric(dim(usNews\_remove\_non\_numeric)[2])

+ for (col in 1:ncol(usNews\_remove\_non\_numeric)) {

+ colTotal=0

+ colLength=0

+ for (row in 1:nrow(usNews\_remove\_non\_numeric)){

+ if(!is.na(usNews\_remove\_non\_numeric[row, col])){

+ colTotal = colTotal + usNews\_remove\_non\_numeric[row, col]

+ colLength = colLength + 1

+ }#end non-missing entries

+ }

+ colAverages[col]=colTotal/colLength

+ }#end iteration over columns

+ return(colAverages)

+ }#end function "mymean2"

> average3<-mymean2(usNews)

> average3

Which do you expect to be most efficient?  Explain your answer in 1-3 sentences.

The built-in functions will be the fastest. With the loops the function needs to reallocate memory and figure what is a string or a number each time it goes through the loop.

1. Write functions for each of the 3 methods in part c. Apply them to the US News and World Report data and check that all 3 methods give the same answers.

This is just for c iii, but I checked and they are all the same. ci and cii reported the column names and the format was not as readable in this document as the following:

[1] 1.639017 506.837838 461.223938 967.978177 22.120448 462.235751 583.148964 418.487047 530.452073 19.819005 25.113122 2752.097523 1870.683191 778.880493 25.671978

[16] 52.350000 3692.665127 1081.526772 7897.274371 9276.905616 4162.106852 2514.681957 2060.983831 392.012646 549.972887 1389.291704 68.645669 75.231132 14.858769 20.912963

[31] 8987.890736 60.405316 120.913212

e. Use *microbenchmark* to compare the median running time of the methods in part c.  Write 1-3 sentences describing which method is most efficient.

* It’s OK if the result doesn’t match your prediction in part c, as long as your prediction was well-justified. Different people may get different results, depending on details of your code.
* If you get the warning message, “Could not measure a positive execution time,” double-check that you included parentheses at the end of each function call.

The vector calculations were the most efficient. I was expecting mean(x) to be faster than mymean(x) because it used existing functions. Maybe if it ran 1000 more times I’d see the difference I expected. In c iii there were multiple loops which had to reallocate memory and figure out what is what each time it loops thru. It took so long I thought I did something wrong. Which really helped me get how important efficient coding is.

Unit: microseconds

Unit: microseconds

expr min lq mean median uq max neval

mean(x) 2.802 3.855 9.92e+00 13.2 15.1160 23.367 100

mymean(x) 2.018 2.651 7.12e+00 8.2 10.8095 17.960 100

mymean2(x) 956779 991785 1.0 e+06 1003494 1019305 1174526 100

Submit a .docx, .pdf, or .r file to GitHub, containing your R code, R output, and written interpretations and explanations.