hw3\_report

Yihao Zhang

September 19, 2016

# HW 3 - Due Tuesday Sept 20, 2016. Upload R file to Moodle with name: HW3\_490IDS\_YOURNETID.R

# Do Not remove any of the comments. These are marked by

# The .R file will contain your code and answers to questions.

# Name:

# Main topic: Using the "apply" family function

# Q1 (5 pts)

# Given a function below,

myfunc <- function(z) return(c(z,z^2, z^3%/%2))

# (1) Examine the following code, and briefly explain what it is doing.

y = 2:8  
myfunc(y)

## [1] 2 3 4 5 6 7 8 4 9 16 25 36 49 64 4 13 32  
## [18] 62 108 171 256

matrix(myfunc(y),ncol=3)

## [,1] [,2] [,3]  
## [1,] 2 4 4  
## [2,] 3 9 13  
## [3,] 4 16 32  
## [4,] 5 25 62  
## [5,] 6 36 108  
## [6,] 7 49 171  
## [7,] 8 64 256

### Your explanation

This is generating a matrix with 3 columns. First column will be y. Second column will be y^2. Third column will be (y^3)/2. #(2) Simplify the code in (1) using one of the "apply" functions and save the result as m. ###code & result

m = matrix(c(y,apply(matrix(c(y)), 2, function(x) x^2),apply(matrix(c(y)), 2, function(x) x^3%/%2)), ncol=3)  
print(m)

## [,1] [,2] [,3]  
## [1,] 2 4 4  
## [2,] 3 9 13  
## [3,] 4 16 32  
## [4,] 5 25 62  
## [5,] 6 36 108  
## [6,] 7 49 171  
## [7,] 8 64 256

# (3) Find the row product of m.

### code & result

apply(m, 1, prod)

## [1] 32 351 2048 7750 23328 58653 131072

print("32 351 2048 7750 23328 58653 131072") #(4) Find the column sum of m in two ways. ###code & result

print("First,use apply")

## [1] "First,use apply"

apply(m,2,sum)

## [1] 35 203 646

print("Second, use matrix multiply")

## [1] "Second, use matrix multiply"

temp <- matrix(rep(1, times = 8-2+1),nrow=1,ncol=8-2+1)  
temp%\*%m

## [,1] [,2] [,3]  
## [1,] 35 203 646

# (5) Could you divide all the values by 2 in two ways?

### code & result

print("First,use apply")

## [1] "First,use apply"

apply(m,1:2,function(x) x/2)

## [,1] [,2] [,3]  
## [1,] 1.0 2.0 2.0  
## [2,] 1.5 4.5 6.5  
## [3,] 2.0 8.0 16.0  
## [4,] 2.5 12.5 31.0  
## [5,] 3.0 18.0 54.0  
## [6,] 3.5 24.5 85.5  
## [7,] 4.0 32.0 128.0

print("Second, use matrix multiply")

## [1] "Second, use matrix multiply"

temp = diag(8-2+1)/2  
temp%\*%m

## [,1] [,2] [,3]  
## [1,] 1.0 2.0 2.0  
## [2,] 1.5 4.5 6.5  
## [3,] 2.0 8.0 16.0  
## [4,] 2.5 12.5 31.0  
## [5,] 3.0 18.0 54.0  
## [6,] 3.5 24.5 85.5  
## [7,] 4.0 32.0 128.0

# Q2 (8 pts)

# Create a list with 2 elements as follows:

l <- list(a = 1:10, b = 11:20)

# (1) What is the product of the values in each element?

lapply(l, prod)

## $a  
## [1] 3628800  
##   
## $b  
## [1] 670442572800

# (2) What is the (sample) variance of the values in each element?

lapply(l,var)

## $a  
## [1] 9.166667  
##   
## $b  
## [1] 9.166667

# (3) What type of object is returned if you use lapply? sapply? Show your R code that finds these answers.

typeof(lapply(l,var))

## [1] "list"

typeof(sapply(l,var))

## [1] "double"

# Now create the following list:

l.2 <- list(c = c(21:30), d = c(31:40))

# (4) What is the sum of the corresponding elements of l and l.2, using one function call?

mapply(function(x,y) x+y,x=l,y=l.2)

## a b  
## [1,] 22 42  
## [2,] 24 44  
## [3,] 26 46  
## [4,] 28 48  
## [5,] 30 50  
## [6,] 32 52  
## [7,] 34 54  
## [8,] 36 56  
## [9,] 38 58  
## [10,] 40 60

# (5) Take the log of each element in the list l:

lapply(l,function(x) log(x))

## $a  
## [1] 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379 1.7917595 1.9459101  
## [8] 2.0794415 2.1972246 2.3025851  
##   
## $b  
## [1] 2.397895 2.484907 2.564949 2.639057 2.708050 2.772589 2.833213  
## [8] 2.890372 2.944439 2.995732

# (6) First change l and l.2 into matrixes, make each element in the list as column,

### your code here

l = matrix(unlist(l), ncol = 2)  
l.2 = matrix(unlist(l.2), ncol = 2)

# Then, form a list named mylist using l,l.2 and m (from Q1) (in this order).

### your code here

mylist = list(l,l.2,m)

# Then, select the first column of each elements in mylist in one function call (hint '[' is the select operator).

### your code here

lapply(mylist,'[',,1)

## [[1]]  
## [1] 1 2 3 4 5 6 7 8 9 10  
##   
## [[2]]  
## [1] 21 22 23 24 25 26 27 28 29 30  
##   
## [[3]]  
## [1] 2 3 4 5 6 7 8

# Q3 (3 pts)

# Let's load our friend family data again.

load(url("http://courseweb.lis.illinois.edu/~jguo24/family.rda"))

# (1) Find the mean bmi by gender in one function call.

apply(matrix(c(family$bmi)),2,mean)

## [1] 24.57612

# (2) Could you get a vector of what the type of variables the dataset is made of？

sapply(family,class)

## firstName gender age height weight bmi overWt   
## "factor" "factor" "integer" "numeric" "integer" "numeric" "logical"

# (3) Could you sort the firstName in height descending order?

lapply(list(family[order(family$height,decreasing = TRUE),1]),'[',1:length(family$firstName))

## [[1]]  
## [1] Joe Tom Tom Liz Jon Tim Bob Ann Dan Art Sal May Sue Zoe  
## Levels: Ann Art Bob Dan Joe Jon Liz May Sal Sue Tim Tom Zoe

# Q4 (2 pts)

# There is a famous dataset in R called "iris." It should already be loaded

# in R for you. If you type in ?iris you can see some documentation. Familiarize

# yourself with this dataset.

# (1) Find the mean petal length by species.

### code & result

by(iris$Petal.Length, iris$Species, mean)

## iris$Species: setosa  
## [1] 1.462  
## --------------------------------------------------------   
## iris$Species: versicolor  
## [1] 4.26  
## --------------------------------------------------------   
## iris$Species: virginica  
## [1] 5.552

# (2) Now obtain the sum of the first 4 variables, by species, but using only one function call.

### code & result

by(iris[,1:4],iris$Species,sum)

## iris$Species: setosa  
## [1] 507.1  
## --------------------------------------------------------   
## iris$Species: versicolor  
## [1] 714.6  
## --------------------------------------------------------   
## iris$Species: virginica  
## [1] 857

# Q5 (2 pts)

# Below are two statements, their results have different structure,

lapply(1:4, function(x) x^3)

## [[1]]  
## [1] 1  
##   
## [[2]]  
## [1] 8  
##   
## [[3]]  
## [1] 27  
##   
## [[4]]  
## [1] 64

sapply(1:4, function(x) x^3)

## [1] 1 8 27 64

# Could you change one of them to make the two statements return the same results (type of object)?

as.numeric(lapply(1:4, function(x) x^3),nrow=1)

## [1] 1 8 27 64

# Q6. (5 pts) Using the family data, fit a linear regression model to predict

# weight from height. Place your code and output (the model) below.

line = lm(family$weight ~ family$height)  
print("output is Coefficients:  
 (Intercept) family$height   
 -455.666 9.154 ")

## [1] "output is Coefficients:\n (Intercept) family$height \n -455.666 9.154 "

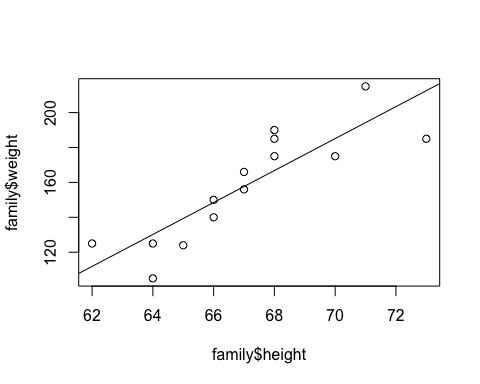
# How do you interpret this model?

print("The coefficient is positive, it means that height and weight has positive relationship,which means when height increases, weight tends to increse as well.")

## [1] "The coefficient is positive, it means that height and weight has positive relationship,which means when height increases, weight tends to increse as well."

# Create a scatterplot of height vs weight. Add the linear regression line you found above.

plot(family$height,family$weight,type = "p")  
abline(line)



# Provide an interpretation for your plot.

print("This is the linear regression lien generated from weight and height. As we can see the weight tends to increase when height increases")

## [1] "This is the linear regression lien generated from weight and height. As we can see the weight tends to increase when height increases"