#Exercise 1 Biostatistics Katherine Barrett

1/23/17

**#Question 1:**

> planet <- c('Mercury', 'Venus', 'Earth', 'Mars', 'Jupiter', 'Saturn', 'Uranus', 'Neptune') #making a list of planet names

#pseudocode = map out how you might take steps to translate into real code

> circumference.miles <- c(9525,23628,24900,13264,279118,235299,99786,96692) # making list of circumference in miles

> circumference.km <- circumference.miles/0.621 # converting circumference to km, but could also use sapply function

> sapply(vector name, function)

> pi <- 3.14159

> 2pi <- pi\*2 # storing 2pi as a variable for when I calculate diameter

> twopi <- pi\*2

> diameter <- (circumference.km/twopi)\*2

# other group did data$diameter <- data$circ/0.621\*pi # R knows to apply this arithmetic manipulation to all elements

# also could make one vector of diameters, then use names (name of vector) to apply planet names for each diameter

> Planet.data <- data.frame(planet,circumference.km,diameter, stringsAsFactors = FALSE) #can make dataframe before or after calculating circumference

> Planet.data

planet circumference.km diameter.km

1 Mercury 15338.16 4882.293

2 Venus 38048.31 12111.163

3 Earth 40096.62 12763.161

4 Mars 21359.10 6798.818

5 Jupiter 449465.38 143069.394

6 Saturn 378903.38 120608.794

7 Uranus 160685.99 51147.982

8 Neptune 155703.70 49562.070

**#Question 2**: In 3 different ways, create and print a matrix named "B" that contains 7 columns and 5 rows with all the

# elements containing the value 7

# Approach 1

B <- matrix(7, nrow = 5, ncol = 7)

> B

[,1] [,2] [,3] [,4] [,5] [,6] [,7]

[1,] 7 7 7 7 7 7 7

[2,] 7 7 7 7 7 7 7

[3,] 7 7 7 7 7 7 7

[4,] 7 7 7 7 7 7 7

[5,] 7 7 7 7 7 7 7

#Approach 2: make an array?

> B = array(rep(7), dim = c(5,7)) # telling R to repeat 7 as the value to fill all elements, with dimensions 5 rows and 7 columns

> B

[,1] [,2] [,3] [,4] [,5] [,6] [,7]

[1,] 7 7 7 7 7 7 7

[2,] 7 7 7 7 7 7 7

[3,] 7 7 7 7 7 7 7

[4,] 7 7 7 7 7 7 7

[5,] 7 7 7 7 7 7 7

#Approach 3: matrix of 7 rows and 5 columns, then transpose

B <- matrix(7, nrow = 7, ncol = 5)

> t(B) # transpose

[,1] [,2] [,3] [,4] [,5] [,6] [,7]

[1,] 7 7 7 7 7 7 7

[2,] 7 7 7 7 7 7 7

[3,] 7 7 7 7 7 7 7

[4,] 7 7 7 7 7 7 7

[5,] 7 7 7 7 7 7 7

# could also do: v = rep(7,5)

B = cbind(v,v,v,v,v,v,v)

# And what if you do not just want to repeat the same value?

Matrix(1:7, 5, 7)

V = rep(7,7) #gives seven 7s

**#Question 3:** Data table into excel and save as variable M.

#3a. sorted <- M[order(M$V1),]

> sorted

V1 V2

1 1998 10.0

2 1999 12.5

3 2000 15.6

4 2001 19.5

5 2002 24.4

6 2003 NA

7 2004 36.2

8 2005 27.7

> sorted <- M[order(M$V1),]

**# 3b.** Calculate mean of second column

M = read.csv(“exercise1.csv or whatever blah blah”, header = FALSE)

mean3 <- mean(M[,2], na.rm = TRUE) # could also use colMeans to get mean of all columns

> mean = 20.84286

#This worked!

**#3c.** Reverse the order of rows so most recent year and data first

> M[order(-M$V1),] # needed to use -M$V1 because the - indicates descending order of years

# could also use apply function and rev function to a new vector

# could also do revM <- M[order(M$V1, decreasing = TRUE), probably want to go this route, scoping can occur with the - sign

V1 V2

8 2005 27.7

7 2004 36.2

6 2003 NA

5 2002 24.4

4 2001 19.5

3 2000 15.6

2 1999 12.5

1 1998 10.0

**#3d.** Find year that had lowest value in 2nd column

> lowest.value <- min(M$V2, na.rm = TRUE)

> lowest.value

# could do (M[which.min(M[,2])])

(colnames(M) = c(“Year”, “Value”))

(M[which.min

# apply(M,2,min,na.rm=TRUE)

[1] 10

**#3e.** Calculate the median of the second column for data collected in 2000 or later

ques3e <- subset(M, M$V1 >= 2000) # subsetting using subset function, need $ to indicate how we are subsetting

> ques3e

V1 V2

3 2000 15.6

4 2001 19.5

5 2002 24.4

6 2003 NA

7 2004 36.2

8 2005 27.7

> ques3emed <- median(ques3e[,2], na.rm = TRUE)

> ques3emed

# or could do: (median3 = median(M$Value[which(M$year>= 2000)], na.rm = TRUE)

[1] 24.4

**4a.**

**Could also do: groupData <- list(ages = c(), names = c()…)**

group.dataframe <- data.frame(ages, names, MvF, stringsAsFactors=FALSE)

> group.dataframe

ages names MvF

1 22 Colin Male

2 26 Katie Female

3 21 Erin Female

4 26 Whitney Female

> female.mean <- mean(ages\*2)

> female.mean

[1] 47.5

# could also do: mean(groupData$ages[which(groupData$MvF==”F”)]\*2)

**#4b.**

males <- subset(group.dataframe, group.dataframe$MvF == "Male")

> males

ages names MvF

1 22 Colin Male

**#4c.** Age of group member with name that is last in alphabetical order

hmmm, not doing this right

# groupData$ages[which(groupData$names==tail(sort(groupData$name), n=1))]

#tail function gives the last item in a list, and n = # tells R how many last n names you want

> group.dataframe

ages names MvF

1 22 Colin Male

2 26 Katie Female

3 21 Erin Female

4 26 Whitney Female

> ques5 <- sort(group.dataframe, group.dataframe$names, decreasing = TRUE)

Error in order(c(2L, 3L, 1L, 3L, 5L, 10L, 6L, 12L, 11L, 7L, 7L, 7L), na.last1 = "Colin", :

argument lengths differ

**#5.**

Lake.exercise1 <- read.csv("C:/Users/klbai/Desktop/Biocomputing SPR 2017/Lake.exercise1.csv")

> View(Lake.exercise1)

> lakedata <- Lake.exercise1

> lake.CV <- apply(lakedata[,3:7], 2, FUN = function(i) sd(i)/mean(i)) #This is the way to calculate CV in one line!

> lake.CV

area\_ha TP\_mgm3 DOC\_gm3 R GPP

1.7084902 0.6528732 0.7237857 1.6920424 1.4760205

#Calculate difference between mean TP for northern and southern lakes

#first need to subset dataframe by north and south

NvS <- subset(lakedata, lakedata$northVsouth == "north")

> NvS

Lake northVsouth area\_ha TP\_mgm3 DOC\_gm3 R GPP

3 brown north 32.9 55.9 9.1 5.8 4.1

4 crampton north 25.8 13.2 4.0 12.4 10.6

6 kickapoo north 7.9 34.9 14.2 119.8 75.3

8 ward north 2.7 28.1 7.0 28.3 22.7

#could use waterN <- ddply(water, NorthvSout), summarize, mean = mean(water…) see the plyr package and ddply function

north.TPmean <- mean(NvS[,4])

> north.TPmean

[1] 33.025

> NvS2 <- subset(lakedata, lakedata$northVsouth == "south")

> south.TPmean <- mean(NvS2[,4])

> south.TPmean

[1] 17.86667

diff <- north.TPmean - south.TPmean

> diff

[1] 15.15833