Charlee Cobb - Transcriptomics, Homework 1

netID: cac9995

Charlee Cobb

PART 1 **1. What is an R object? (1pt)** An R object is an instance of either an array, a matrix, a dataframe, a vector, or a list that holds a value of data.

2. How many ways can data be assigned to an R object? (1pt) Data can be assigned to an R object by using the "=" symbol, or the "<-" symbol.

3. Why do you think it is important to ensure that data objects are of the correct type? (1pt) It's important to ensure that data objects are of the correct type because the type of data limits the number of ways functions can manipulate the object, and limit ways objects interact with each other. For example, if you have a vector of character represented numbers and a vector of numericals, you couldn't run any operations on the two vectors at once because they have different data types. One vector is an object pf characters, and one is an object of numericals.

4. What is the relationship between vectors, matrices, and data frames? (1pt) Vectors, matrices, and data frames all hold data points or values. Vectors are one deminsional, and only hold values that are all the same type(ie characters or numerical values). A matrix is made from 2 or vectors that have been coerced together. However, the vectors building the matirx all have to have the same data type. A dataframe is preferred over a matrix when you want to store values that are different data types (ie. numericals and characters together). However for both matrices and data frames, the coerced vectors must always be of the same length to ensure the same number of rows and columns in either the matrix or data frame.

5. Why might a data frame be more suitable than a matrix for holding heterogeneous biological data? (1pt) As stated above, a data frame is preferred over a matrix when you want to store values that are different data types. With a data frame, the heterogeneous biological data can be stored because the data frame accepts numerical values, characters, and other data types to remain in the data frame without converting them. So data with gene name, expression value, reference id, and annotations can all be stored in a data frame. A matrix wouldn't allow the expression value to be stored, unless the values were converted to character format.

PART 2 **1. Create a matrix (call it transcriptome) with the values below. The experiments are column names and genes are the row names. (3pts)**

```
##creating matrix##

#make vectors of each gene by row:
GeneA <- c(89,78,77,56)
GeneB <- c(90,99,85,97)
GeneC <- c(78,94,99,87)
GeneD <- c(81,83,80,79)
GeneE <- c(62,51,99,88)

#combine above vectors into a matrix named transcriptome. Specify number of rows
#and columns, use 'byrow' condtion set to TRUE to ensure data from vectors are entered by row
transcriptome <- matrix(c(GeneA, GeneB, GeneC, GeneD, GeneE), nrow=5, byrow=TRUE, ncol= 4)
```

```
#add names and columns to the using the transcriptome matrix rownames, colnames function
rownames(transcriptome) <- c("GeneA", "GeneB", "GeneC", "GeneD", "GeneE")
colnames(transcriptome) <- c("Control", "Nitrogen", "Phospate", "Potassium")
print(transcriptome)
```

```
##       Control Nitrogen Phospate Potassium
## GeneA      89       78       77        56
## GeneB      90       99       85        97
## GeneC      78       94       99        87
## GeneD      81       83       80        79
## GeneE      62       51       99        88
```

2. Use R code to calculate the average expression for each gene across all experiments. Call the vector (call it expression_average). (The vector should contain 5 values – one for each gene) (3pts)

```
##make vector of average expression##

#using the apply function, we are going across each row(1) in the data set
#transcriptome, and using the built in mean function to find the average of each
#gene's expression data
expression_average <- apply(transcriptome, 1, mean)
print(expression_average)
```

```
## GeneA GeneB GeneC GeneD GeneE
## 75.00 92.75 89.50 80.75 75.00
```

3. Sort the matrix so that the gene with the highest average expression value is on top and save it into a new data frame (call it "sorted_genes") (4pts)

```
##sort genes by row and put into a dataframe.##

sorted_genes <-as.data.frame(transcriptome[order(apply(transcriptome, 1, mean), decreasing = T),])
sorted_genes
```

```
##       Control Nitrogen Phospate Potassium
## GeneB      90       99       85        97
## GeneC      78       94       99        87
## GeneD      81       83       80        79
## GeneA      89       78       77        56
## GeneE      62       51       99        88
```

```
#there is an option to make a separate matrix of sorted data and then convert it to
#a data frame, but we can do that in one line here.
#starting with the order() function, we take the apply function we used above to
#find a list of means, then use decreasing = T to say we want to order to go from
#largest value at the top to smallest value. Putting this inside transcriptome[]
#means we are applying this order change to the transcriptome matrix. Finally,
#we put this all inside the as.data.frame() function to save the changes in the
#data frame sorted genes.
```

***PART 3* a. Load the file expvalues.txt into R. (2pts)**

```
#this file is saved in my project directory, so no path is added here
expvalues <- read.table("expvalues.txt")
head(expvalues)
```

```
##            Control1  Control2  Control3 Treatment1 Treatment2 Treatment3
## 244901_at 229.98565 353.59949 178.49171  112.90800  152.91835  320.83235
## 244902_at 171.14980  84.45094  41.37195  170.17262  134.40814  193.52611
## 244903_at 314.97768 373.31250  52.90873  196.30256  237.51520  253.37774
## 244904_at  24.04366 175.31604  94.68424   82.78488   18.43639   87.00857
## 244905_at  15.86923  40.04125  58.76573  341.35340   16.80135   68.94204
## 244906_at 381.06379 218.90597 307.01488  162.54109  264.47945  263.66098
```

b. Calculate the mean of the control samples and the mean of the treatment samples for each gene.(You can use a loop or apply functions) (5pts)

```
##get control average and treatment average per gene##

#used this line to make sure I was pulling a specific amount of columns:
#print(head(expvalues[,c(1:3)]))

#(expvalues[,c(1:3)] and (expvalues[,c(4:6)] subset the expvalue data frame to
#control columns only and treatment columns only, respectively.

#using the apply function to find the mean of each gene (per row, 1) in the subset
#data frame of expvalues
control.means <- apply(expvalues[,c(1:3)], 1, mean)
head(control.means)
```

```
## 244901_at 244902_at 244903_at 244904_at 244905_at 244906_at
## 254.02562  98.99090 247.06630  98.01465  38.22541 302.32821
```

```
treatment.means <- apply(expvalues[,c(4:6)], 1, mean)
head(treatment.means)
```

```
## 244901_at 244902_at 244903_at 244904_at 244905_at 244906_at
## 195.55290 166.03562 229.06517  62.74328 142.36560 230.22717
```

```
#expected means of the 244901_at gene expression to verify results:
#control:
print("control:")
```

```
## [1] "control:"
```

```
mean(c(229.98565, 353.59949, 178.49171))
```

```
## [1] 254.0256
```

```r
#treatment:
print("treatment")
```

```
## [1] "treatment"
```

```r
mean(c(112.90800, 152.91835, 320.83235))
```

```
## [1] 195.5529
```

Calculate the fold change for each gene (fold change is the ratio of average treatment to average control). (2pts)

```r
##calculate fold change##
#fold change == average treatment to average control

#make a data frame of the average treatment values and average control values:
average.expvalues <- matrix(c(treatment.means, control.means), ncol = 2)

#get row names from expvalues and add them to average.expvalues
rownames(average.expvalues) <- c(row.names(expvalues))

#Set column names to treatment and control
colnames(average.expvalues) <- c("treatment", "control")
head(average.expvalues)
```

```
##           treatment   control
## 244901_at 195.55290 254.02562
## 244902_at 166.03562  98.99090
## 244903_at 229.06517 247.06630
## 244904_at  62.74328  98.01465
## 244905_at 142.36560  38.22541
## 244906_at 230.22717 302.32821
```

```r
#divide column 1(treatment) by column 2(control)
gene.fold.change <- average.expvalues[,1]/average.expvalues[,2]
head(gene.fold.change)
```

```
## 244901_at 244902_at 244903_at 244904_at 244905_at 244906_at
## 0.7698157 1.6772817 0.9271405 0.6401419 3.7243712 0.7615140
```

```r
#expected fold change of the 244901_at gene expression to verify results:
#control: 254.0256
#treatment: 195.5529
195.5529 / 254.0256
```

```
## [1] 0.7698157
```

d. Take the log2 of the fold change. You have just calculated log fold change(LFC).(2pts)

```
##calculate log fold change##

#use the log2() function to apply log2 to each value in the gene.fold.change object
gene.log.fold.change <- log2(gene.fold.change)
head(gene.log.fold.change)
```

```
##  244901_at  244902_at  244903_at  244904_at  244905_at  244906_at
## -0.3774151  0.7461250 -0.1091402 -0.6435364  1.8969969 -0.3930575
```

e. How many genes have a LFC > 1 OR < -1 ? (2pts) There are 1850 genes with a LFC greater than(>) 1, and 2394 with an LFC less than (<) -1

```
##find genes with values >1 and values < -1

#> 1
LFC.greater <- gene.log.fold.change[gene.log.fold.change > 1]
head(LFC.greater)
```

```
##   244905_at   244914_at   244916_at   244917_at   244945_at 244956_s_at
##    1.896997    1.208409    1.069662    1.002224    1.007064    1.814538
```

```
length(LFC.greater) #length shows number of values (genes) in the LFC.greater object
```

```
## [1] 1850
```

```
#< -1
LFC.less <- gene.log.fold.change[gene.log.fold.change < -1]
length(LFC.less)
```

```
## [1] 2394
```

f. Save the names of the genes that have a LFC > 1 into a file called "Induced_genes.txt" (2pts)

```
##save the names of the genes in LFC.greater

#the LFC.greater object has the gene names as an attribute under $names, so when
#the object is printed, we see the gene names listed

#extract the gene names from the attribute using the attribute function
#?attribute()
LFC.greater.attr <- attributes(LFC.greater)
head(LFC.greater.attr$names)
```

```
## [1] "244905_at"   "244914_at"   "244916_at"   "244917_at"   "244945_at"
## [6] "244956_s_at"
```

```
#write the gene names in the file called "Induced_genes.txt" with the write()
#function
write((LFC.greater.attr$names), file = "Induced_genes.txt")
```

Using the same set of induced genes in the previous question, create a boxplot to show the distribution of values for each induced gene in each experiment. The x-axis should have all six experiments, and the y-axis is the expression level. Save the boxplot as a pdf file called "boxplot.pdf"(5pts)

```
##make a box plot of induced genes##

#subset the data from expvalues to induced.gene.expvalues so only the induced
#genes from the above chunk are present. Using the %in% pipe to subset data

induced.gene.expvalues <- expvalues[rownames(expvalues) %in% LFC.greater.attr$names, ]
colnames(induced.gene.expvalues) <- c("Control1", "Control2", "Control3", "Treatment1", "Treatment2", "
head(induced.gene.expvalues)
```

```
##               Control1  Control2 Control3 Treatment1 Treatment2 Treatment3
## 244905_at    15.869235  40.04125 58.76573  341.35340  16.801354   68.94204
## 244914_at     9.206320  23.66153 37.63685   25.38165   7.404548  130.13789
## 244916_at     3.371939  38.96267 16.94395   68.91271   2.800184   52.70934
## 244917_at    12.290308  16.83069 19.14878   48.15070  10.206957   38.33083
## 244945_at   157.299361 204.15104 24.23849  552.28230  93.524026  129.35776
## 244956_s_at  95.024495  40.62732 28.03325  459.47534  92.258511   24.02345
```

```
#make a pdf
pdf("boxplot.pdf")
par(mar=c(14, 4.1, 4.1, 2.1))

#make a boxplot with the subseted data above, save it the pdf open
boxplot(induced.gene.expvalues$Control1,
        induced.gene.expvalues$Control2,
        induced.gene.expvalues$Control3,
        induced.gene.expvalues$Treatment1,
        induced.gene.expvalues$Treatment2,
        induced.gene.expvalues$Treatment3,
        main = "Induced Gene Expression Distribution",
        xlab = "experiments", ylab = "expression value",
        show.names = TRUE) #show the distribution of all the 6 measurements

#save plots
dev.off()
```

```
## pdf
##   2
```

```
#check <- induced.gene.expvalues[induced.gene.expvalues > 15000]
#length(check)
```