SP1 Analysis

Orfeas Gkourlias

3/10/2022

3. Exploratory Data Analysis

The data will be need to be extracted from the count file first. Since it's in .tsv format, the seperator is going to be tab based. The inclusion of headers adds an X to the sequence ID's, because R is unable to make headers out of just integers. To counteract this, the colnames will be manually added.

3.1 Loading the data

```
file <- c("..\\data\\GSE152262_RNAseq_Raw_Counts.tsv")
raw_data <- read.table(file, sep = '\t', header = TRUE, row.names = 1)
colnames(raw_data) <- c("case4275", "case4277", "con4279", "con4280", "con4280a", "case4281")
raw_data[1:5,]
```

```
##                 case4275 case4277 con4279 con4280 con4280a case4281
## ENSG00000000003       23       30      11      43       31        8
## ENSG00000000005        0        0       0       0        2        0
## ENSG00000000419      778      910     838     911     1113     1051
## ENSG00000000457      378      438     441     772      738      389
## ENSG00000000460       44       51      58      61       65       28
```

```
dim(raw_data)
```

```
## [1] 58307     6
```

```
str(raw_data)
```

```
## 'data.frame':    58307 obs. of  6 variables:
##  $ case4275: int  23 0 778 378 44 14575 30 54 213 546 ...
##  $ case4277: int  30 0 910 438 51 21109 23 89 206 589 ...
##  $ con4279 : int  11 0 838 441 58 7164 94 105 333 452 ...
##  $ con4280 : int  43 0 911 772 61 11710 151 77 419 407 ...
##  $ con4280a: int  31 2 1113 738 65 11846 148 69 384 373 ...
##  $ case4281: int  8 0 1051 389 28 27759 68 50 180 561 ...
```

The data is now loaded in as a data frame. Every row shows the raw counts of a specific gene being expressed. 4275, 4277 and 4281 are the variant types. The datatypes are correct in this case. There should only be integers included, except for the gene names.

Now that the data has been properly loaded, objects can be made to differentiate the control and case counts. Before separating the groups, it'll be useful to apply a log2 function to our data. This makes it so that the data is more informative and tidier, because of outliers and the big range being worked with.

```
raw_data <- log2(raw_data + 1)

case <- raw_data[,c(1,2,6)]
```

```
control <- raw_data[,c(3:5)]

case[1,]
```

```
##                 case4275 case4277 case4281
## ENSG00000000003 4.584963 4.954196 3.169925
```

```
control[1,]
```

```
##                  con4279  con4280 con4280a
## ENSG00000000003 3.584963 5.459432        5
```

The control and case data is now stored in different variables, as shown above.

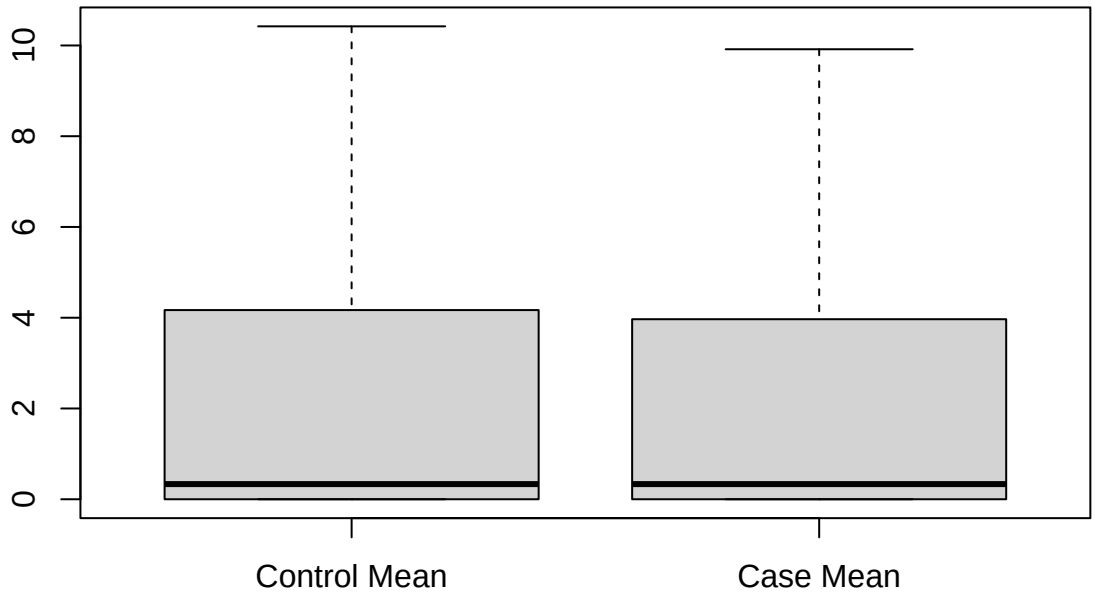
Visualizing using boxplot and density plot

More insight on the data can be gained by plotting and summarizing it. Every column will first be summarized. Following that, the mean values will be compared in a boxplot.

```
summary(raw_data)
```

```
##     case4275          case4277          con4279          con4280
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000
##  Median : 0.000   Median : 0.000   Median : 0.000   Median : 0.000
##  Mean   : 2.424   Mean   : 2.552   Mean   : 2.524   Mean   : 2.579
##  3rd Qu.: 3.907   3rd Qu.: 4.248   3rd Qu.: 4.170   3rd Qu.: 4.248
##  Max.   :23.704   Max.   :23.582   Max.   :23.549   Max.   :23.675
##     con4280a          case4281
##  Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 0.000   1st Qu.: 0.000
##  Median : 0.000   Median : 0.000
##  Mean   : 2.571   Mean   : 2.405
##  3rd Qu.: 4.170   3rd Qu.: 3.907
##  Max.   :24.155   Max.   :23.642
```

```
case$mean = apply(X = case[1:3], MARGIN = 1, FUN = mean)
control$mean = apply(X = control[1:3], MARGIN = 1, FUN = mean)
boxplot(control$mean, case$m, outline = FALSE, names = c("Control Mean", "Case Mean"))
```

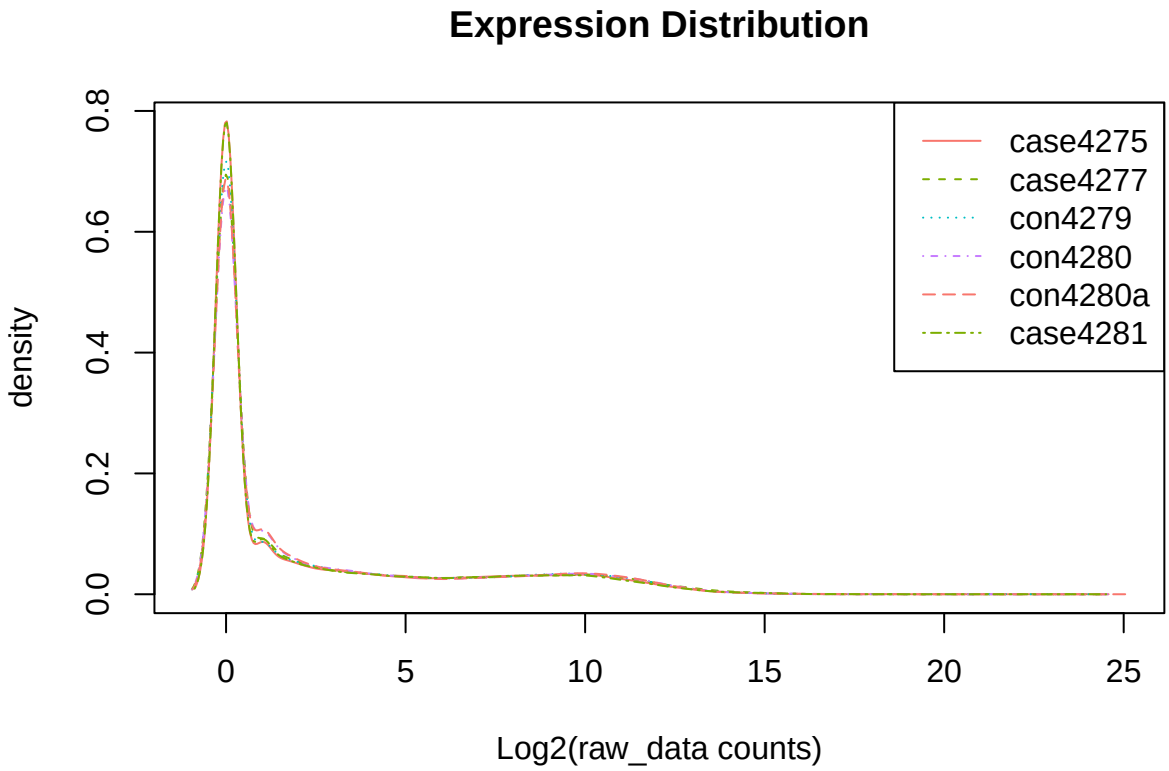


These boxplots are not yet very informative. The only thing that can be seen from them is that the cases have a slightly lower expression level on average

Maybe a density plot allows for a more informative figure.

```
myColors <- hue_pal()(4)
plotDensity(raw_data, col=rep(myColors, each=1), lty=c(1:ncol(raw_data)),
            main = "Expression Distribution", xlab = "Log2(raw_data counts)")

legend('topright', names(raw_data), lty=c(1:ncol(raw_data)),
       col=rep(myColors, each=1))
```



As can be seen in the plot, the highest amount of expressions, besides 0, seem to be around 10.

3.4 Visualizing using heatmap and MDS