Class13

Juan Gonzalez (PID: A69036681)

Transcriptomics and the analysis of RNA-seq data

Bioconductor is a large repository and resource for R packages that focus on analysis of high-throughput genomic data.

The count matrix (called the countData in DESeq2-speak) the value in the i-th row and the j-th column of the data.frame tells us how many reads can be assigned to gene i in sample j. Analogously, for other types of assays, the rows of this matrix might correspond e.g. to binding regions (with ChIP-Seq) or peptide sequences (with quantitative mass spectrometry).

For the sample metadata (i.e. colData in DESeq2-speak) samples are in rows and metadata about those samples are in columns. Notice that the first column of colData must match the column names of countData (except the first, which is the gene ID column) (Figure 2).

```
counts <- read.csv("airway_scaledcounts.csv", row.names=1)
metadata <-  read.csv("airway_metadata.csv")
```

```
head(counts)
```

```
                SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
ENSG00000000003        723        486        904        445       1170
ENSG00000000005          0          0          0          0          0
ENSG00000000419        467        523        616        371        582
ENSG00000000457        347        258        364        237        318
ENSG00000000460         96         81         73         66        118
ENSG00000000938          0          0          1          0          2
                SRR1039517 SRR1039520 SRR1039521
ENSG00000000003       1097        806        604
ENSG00000000005          0          0          0
ENSG00000000419        781        417        509
ENSG00000000457        447        330        324
ENSG00000000460         94        102         74
ENSG00000000938          0          0          0
```

```r
head(metadata)
```

```
          id     dex celltype     geo_id
1 SRR1039508 control   N61311 GSM1275862
2 SRR1039509 treated   N61311 GSM1275863
3 SRR1039512 control  N052611 GSM1275866
4 SRR1039513 treated  N052611 GSM1275867
5 SRR1039516 control  N080611 GSM1275870
6 SRR1039517 treated  N080611 GSM1275871
```

Q1. How many genes are in this dataset?

38694 genes

```r
dim(counts)
```

```
[1] 38694     8
```

How many 'control' cell lines do we have? 4 controls (obtained from the metadata)

#Toy Differential Gene Expression

Note that the control samples are SRR1039508, SRR1039512, SRR1039516, and SRR1039520. This bit of code will first find the sample id for those labeled control. Then calculate the mean counts per gene across these samples:

```r
control <- metadata[metadata[,"dex"]=="control",]
control.counts <- counts[ ,control$id]
control.mean <- rowSums(  control.counts )/4
head(control.mean)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         900.75            0.00          520.50          339.75           97.25
ENSG00000000938
           0.75
```

Q3. How would you make the above code in either approach more robust? Is there a function that could help here?

```r
inds<-metadata$dex=="control"
control.metadata<-metadata[inds,]
control.counts<-counts[, control.metadata$id]
control.mean <- rowMeans(control.counts)
head(control.mean)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         900.75            0.00          520.50          339.75           97.25
ENSG00000000938
           0.75
```

Q4. Follow the same procedure for the treated samples (i.e. calculate the mean per gene across drug treated samples and assign to a labeled vector called treated.mean)

```r
treated.mean<-rowMeans(counts[,metadata[metadata$dex=="treated", ]$id])
```

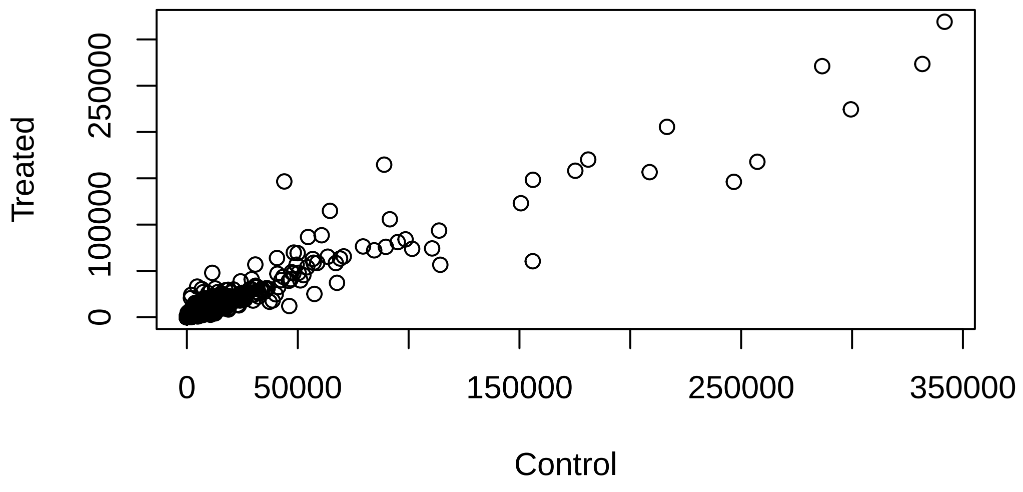
We will now combine our meancount data for bookkeeping purposes

```r
meancounts <- data.frame(control.mean, treated.mean)
```

Always plot your data to get a feel for it

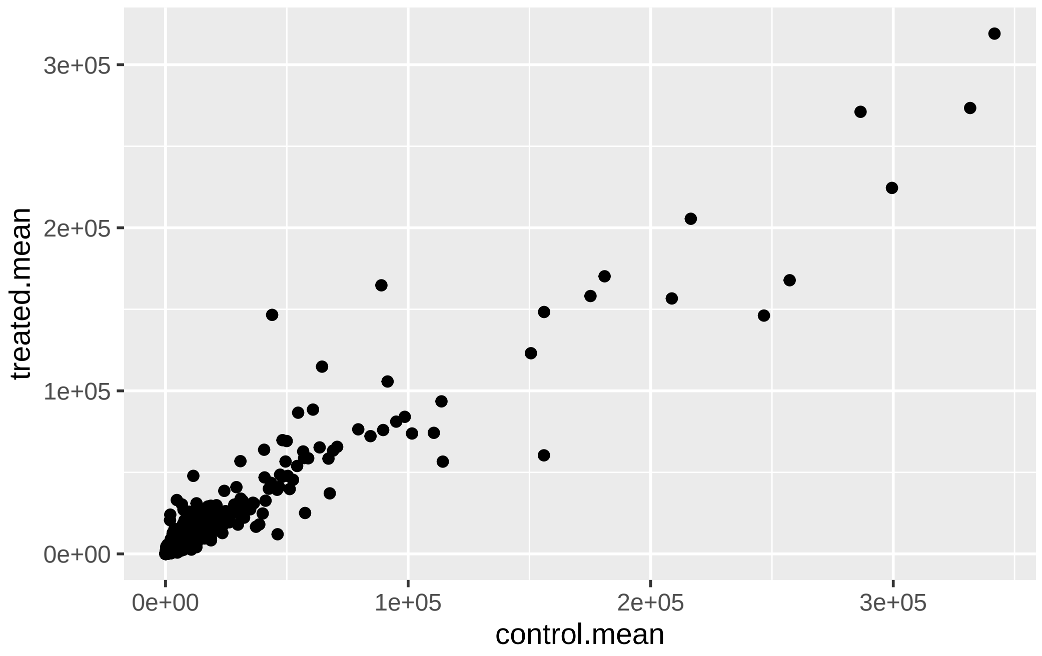
Q5 (a). Create a scatter plot showing the mean of the treated samples against the mean of the control samples. Your plot should look something like the following.

```r
plot(meancounts[,1],meancounts[,2], xlab="Control", ylab="Treated")
```



Q5 (b).You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot?

```
library(ggplot2)
ggplot(meancounts) +
  aes(control.mean, treated.mean) +
  geom_point()
```

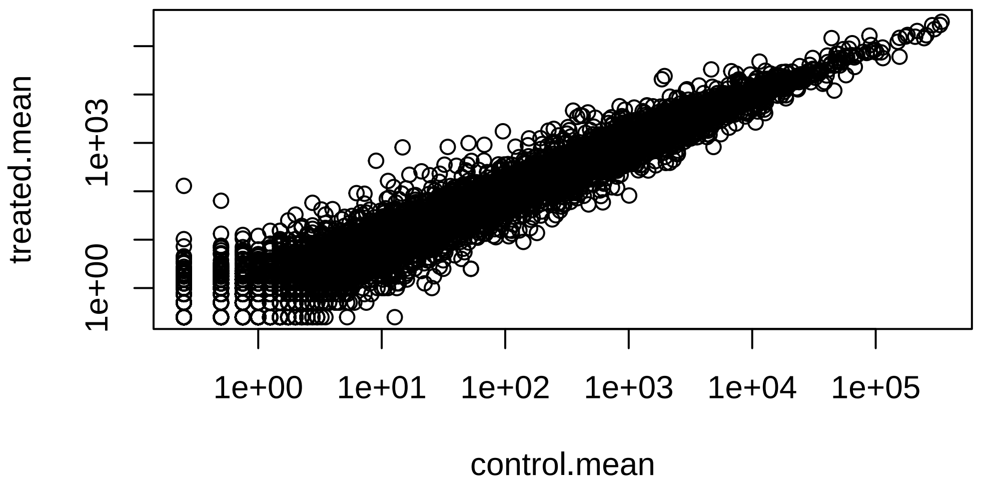


Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

```
plot(meancounts[,1:2], log="xy")
```

```
Warning in xy.coords(x, y, xlabel, ylabel, log): 15032 x values <= 0 omitted
from logarithmic plot
```

```
Warning in xy.coords(x, y, xlabel, ylabel, log): 15281 y values <= 0 omitted
from logarithmic plot
```



We can use `log2()` to ensure that no changes are 0 instead of 1.

```
meancounts$log2fc<-log2(meancounts$treated.mean/meancounts$control.mean)
```

A common rule of thumb in the field is to focus initially on big changes with a cutoff log2fc of +2 or -2.

There are a couple of "weird" results. Namely, the NaN ("not a number") and -Inf (negative infinity) results.

```
zero.vals <- which(meancounts[,1:2]==0, arr.ind=TRUE)

to.rm <- unique(zero.vals[,1])
mycounts <- meancounts[-to.rm,]
head(mycounts)
```

```
                control.mean treated.mean      log2fc
ENSG00000000003       900.75       658.00 -0.45303916
ENSG00000000419       520.50       546.00  0.06900279
ENSG00000000457       339.75       316.50 -0.10226805
ENSG00000000460        97.25        78.75 -0.30441833
ENSG00000000971      5219.00      6687.50  0.35769358
ENSG00000001036      2327.00      1785.75 -0.38194109
```

Q7. What is the purpose of the arr.ind argument in the which() function call above? Why would we then take the first column of the output and need to call the unique() function?

The arr.ind argument will clause `which()` to returb both the row and column indices where there are true values. We put ==0, so this will tell use which genes and samples have zero counts and then ignore the zero count genes. Calling `unique()` will make sure we do not call any row twice if it has a zero in both samples.

Q8. Using the up.ind vector above can you determine how many up regulated genes we have at the greater than 2 fc level?

```r
up.ind <- mycounts$log2fc > 2
sum(up.ind)
```

```
[1] 250
```

Q9. Using the down.ind vector above can you determine how many down regulated genes we have at the greater than 2 fc level?

```r
down.ind <- mycounts$log2fc < (-2)
sum(down.ind)
```

```
[1] 367
```

Q10. Do you trust these results? Why or why not?

We can not trust these result just yet because we need to make sure they are statistically significant.