# Class 12: Differential Expression Analysis

Michelle Woo

## 1. Bioconductor and DESeq2 Setup

First, installing Bioconductor using:

```
# install.packages("BiocManager")
# BiocManager::install()

# BiocManager::install("DESeq2")
```

## 2. Importing countData and colData

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

Viewing each dataset:

```
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
```

```
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?***

```
nrow(counts)
```

```
[1] 38694
```

38694 genes are in the data set.

***Q2. How many control cell lines do we have?***

```
table(metadata$dex)['control']
```

```
control
      4
```

There are 4 control and 4 treated cell lines.

## 3. Toy differential gene expression

We need to find the sample `id` for labeled controls. The average count per gene is then calculated:

```
# filtering out samples with control
control <- metadata[metadata[,"dex"]=="control",]

# ids of the control
control.counts <- counts[ ,control$id]

# finding the average
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
```

***Q3. How would you make the above code in either approach more robust?***

Instead of using the `rowSums` then dividing by 4, we could use `rowMeans` instead. If another dataset is applied and it doesn't have 4 total, then that code will break so it's not applicable across the board. Using `rowMeans` is more general to all datasets.

***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`)***

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

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         658.00            0.00          546.00          316.50           78.75
ENSG00000000938
           0.00
```

Combining the data into a variable:

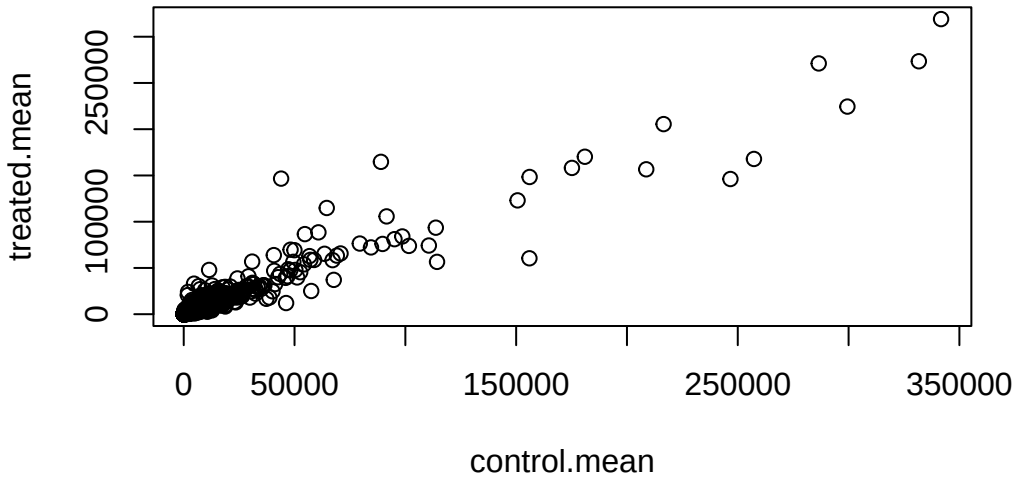
```
meancounts <- data.frame(control.mean, treated.mean)
```

```
# sum of mean counts across all genes for each group
colSums(meancounts)
```

```
control.mean treated.mean
    23005324     22196524
```

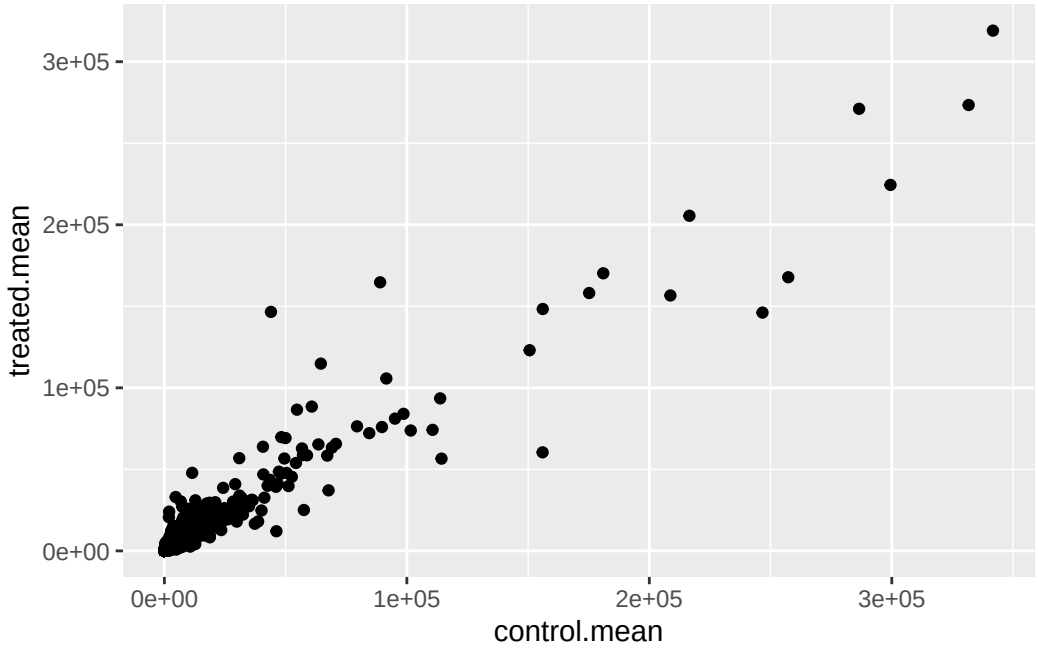
***Q5a. Create a scatter plot showing the mean of the treated samples against the mean of the control samples. Your plot should look something like the following***

```
plot(meancounts)
```



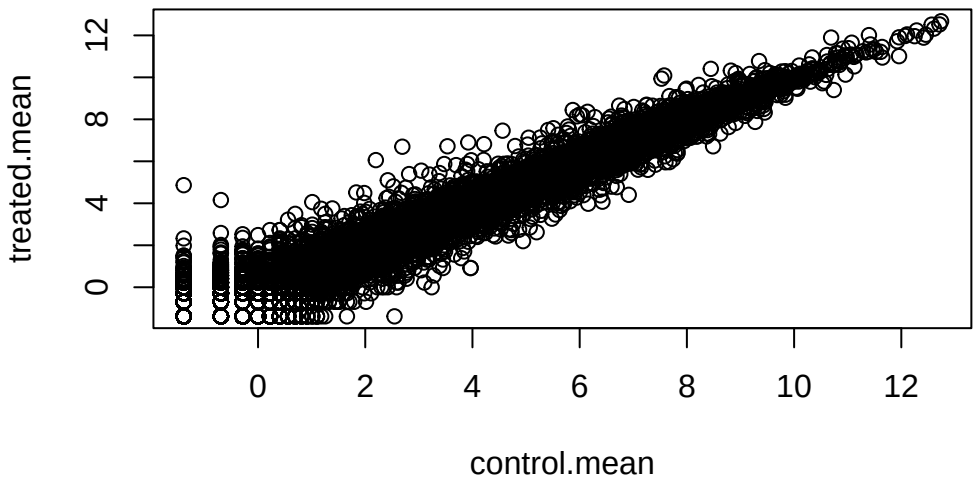

***Q5b. You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot?***

```
library(ggplot2)
ggplot(data = meancounts) +
  geom_point(mapping=aes(x=control.mean, y=treated.mean))
```




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

```
plot(log(meancounts))
```




We can find a gene that has a large change between the control and the treated samples. We can look at the fold change using fold2 to better analyze this.

```
# add log2fc column into dataframe
meancounts$log2fc <- log2(meancounts[,"treated.mean"]/
                          meancounts[,"control.mean"])

head(meancounts)
```

```
                control.mean treated.mean      log2fc
ENSG00000000003       900.75       658.00 -0.45303916
ENSG00000000005         0.00         0.00         NaN
ENSG00000000419       520.50       546.00  0.06900279
ENSG00000000457       339.75       316.50 -0.10226805
ENSG00000000460        97.25        78.75 -0.30441833
ENSG00000000938         0.75         0.00        -Inf
```

For genes with zero expression, we can remove these to streamline our data:

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

to.rm <- unique(zero.vals[,1])

# removing rows with zeroes
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 purpose of the `arr.ind` argument is to get the columns and rows

*Overexpressed and underexpressed genes:*