Class 13

Renny Ng (A98061553)

First, install DESeq2. `install.packages("BiocManager")` `BiocManager::install()` `BiocManager::install("DESeq2")`

Then load up DESeq2 with `library(DESeq2)`

Since everything is an object in R, there are already functions that come from packages (e.g. 'package:stats') and DESeq2 needs to replace some of the pre-existing functions previously in place. DESeq2 will overwrite the functions already there (these can still be called).

##Lab The data for this hands-on session comes from a published RNA-seq experiment where airway smooth muscle cells were treated with dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al. 2014).

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

```
metadata <- read.csv ("airway_metadata.csv")
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
```

There are 38,694 genes.

Q. How many total samples do we have?

```
ncol(counts=="control")
```

```
[1] 8
```

There are 8 samples total.

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

```
ncol(metadata == "control")
```

```
[1] 4
```

There are 4 control cell lines.

Here, let's make sure that our `metadata` id column matches the column names for counts.

```
all(metadata$id == colnames(counts))
```

```
[1] TRUE
```

Our metadata indeed matches our column names.

We have a plan:

- First, extract out the control samples (i.e. the columns)
- Next, calculate the row-wise means (i.e. mean counts for each gene)

First, we need to find the control samples.

Consider the following code: `control <- metadata[metadata[,"dex"]=="control",] control.counts <- counts[ ,control$id] control.mean <- rowSums( control.counts )/4 head(control.mean)`

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

The `apply` function would be helpful here, particularly with the `mean` function.

```
control.inds <- metadata$dex == "control"
control.counts <- counts[, control.inds]
control.mean <- apply(control.counts, 1, mean)
```

> 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.inds <- metadata$dex == "treated"
treated.counts <- counts[, treated.inds]
treated.mean <- apply(treated.counts, 1, mean)
```

Store these together for ease of book-keeping.

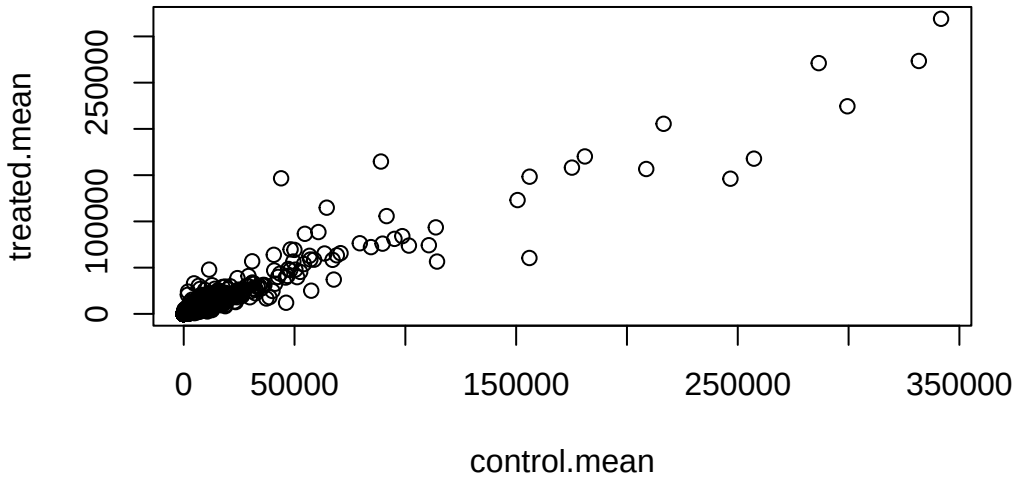
```
meancounts <- data.frame(control.mean, treated.mean)
```

```
colSums(meancounts)
```

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

> Q5 (a). Create a scatter plot showing the mean of the treated samples against the mean of the control samples.

```
plot(control.mean, treated.mean)
```

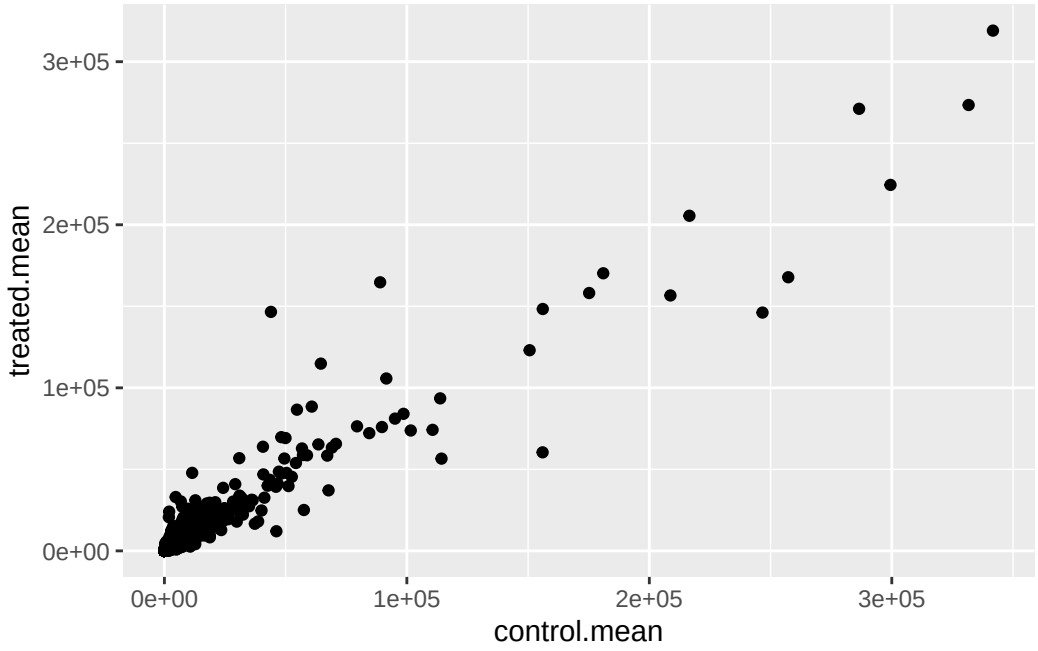


Q5 (b).You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot?

`geom_point()`

```
library(ggplot2)

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



We have such skewed data over a wide range, and we ultimately care about orders of magnitude change anyways. So let's log transform.

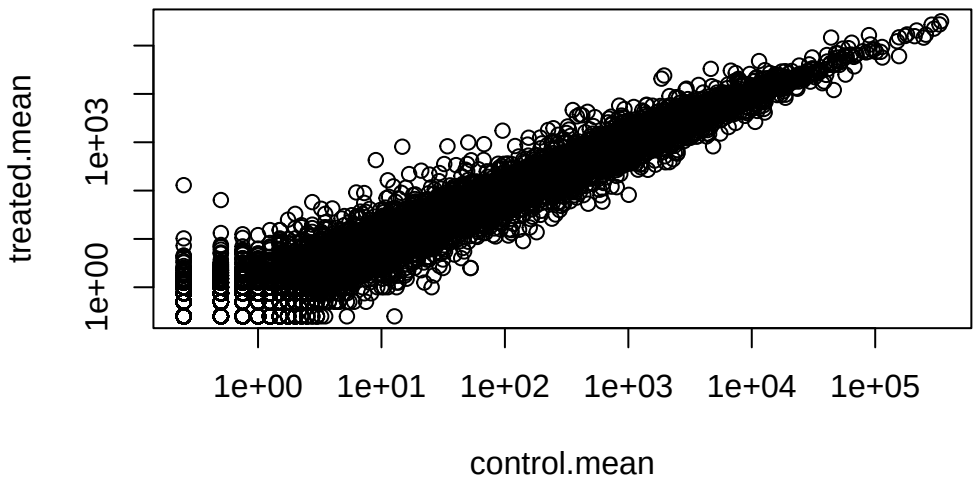
> Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

`log="xy"` to set logarithmic scale to both axes

```
plot(meancounts, 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
```



Genes that are highly expressed in the control are also highly expressed in the treated.

Fold change = measures magnitude differences. We can look at differences here via.a fraction. Divide one condition by another. For example, we can divide Treated/Control.

We often work with log2 transforms. No change will equal 0. Gives the fold-change (double = 1 fold change; half = -1 fold change). We are also mostly interested in very large changes across multiple folds.

Let's calculate the log2 fold change for our treated/control.

```
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
```

We want to get rid of the 0 values, as we can't say anything about the genes. It is common practice to remove/filter out zero count genes. How can we easily find where the 0s are?

```
zero.sums <- rowSums(meancounts[,1:2] == 0)
#Use the trick of logicals being interpreted as numbers
to.rm.ind <- zero.sums > 0
#This allows us to find and exclude the zero sum genes.
mycounts <- meancounts[!to.rm.ind,]
#! changes true to false and false to true
```

```
nrow(mycounts)
```

```
[1] 21817
```

There are 21,817 non-zero genes

A common threshold for calling something "differentially expressed" is a log2 fold-chnage value of +2 or -2.

How many of our remaining genes are upregulated?

> Q8 How many up upregulated genes have a fold-change value of 2 or greater?

```
sum(mycounts$log2fc > +2)
```

```
[1] 250
```

```
sum(mycounts$log2fc >= +2)
```

```
[1] 314
```

Genes with >2 FC: 250 Genes with =>2 FC: 314

> Q9 >Q8 How many up downregulated genes have a fold-change value of -2 or lower?

```
sum(mycounts$log2fc < -2)
```

```
[1] 367
```

```
sum(mycounts$log2fc <= -2)
```

```
[1] 485
```

Genes with <2 FC: 367 Genes with <=2 FC: 485

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

We do not trust these values, because we are only comparing the mean values. They don't account for changes across conditions or samples. The differences in means may not be significant. We need statistics. DESeq will calculate statistics for us, with adjusted corrected P-values with stricter criteria, based on the number of samples run.

```
library(DESeq2)

#| message: false
```

DESeq2 wants our data in a very particular object called a deseq object. We can set this up with functions from within the DESeq2 package. The `DESeq` function is the main analysis function. `DESeqDataSetFromMatrix` is a main function of setting up for DESeq in R.

```
dds <- DESeqDataSetFromMatrix(countData = counts,
                        colData = metadata,
                        design = ~dex)
```

```
converting counts to integer mode
```

```
Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
design formula are characters, converting to factors
```

```
dds <- DESeq(dds)
```

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship

final dispersion estimates

fitting model and testing
```

```
#These DESeq objects have multiple "slots" that can store different types of data.
```

To get the actual results out of this `dds` object, we can use the DESeq function `results()`.

```
res <- results(dds)
head(res)
```

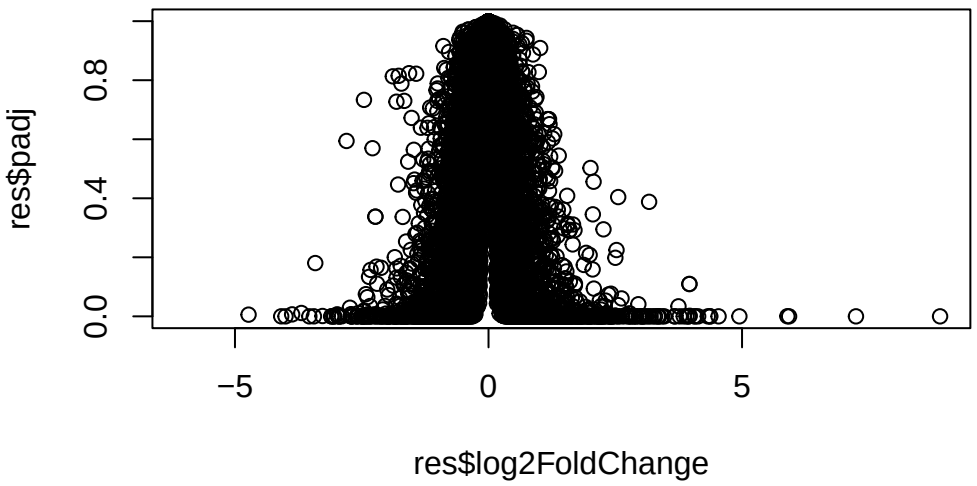
```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 6 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
                 <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
                     padj
                <numeric>
ENSG00000000003  0.163035
ENSG00000000005        NA
ENSG00000000419  0.176032
ENSG00000000457  0.961694
ENSG00000000460  0.815849
ENSG00000000938        NA
```

Note how the `padj` adjusted P-values are stricter than the regular P-values.

##Volcano plot

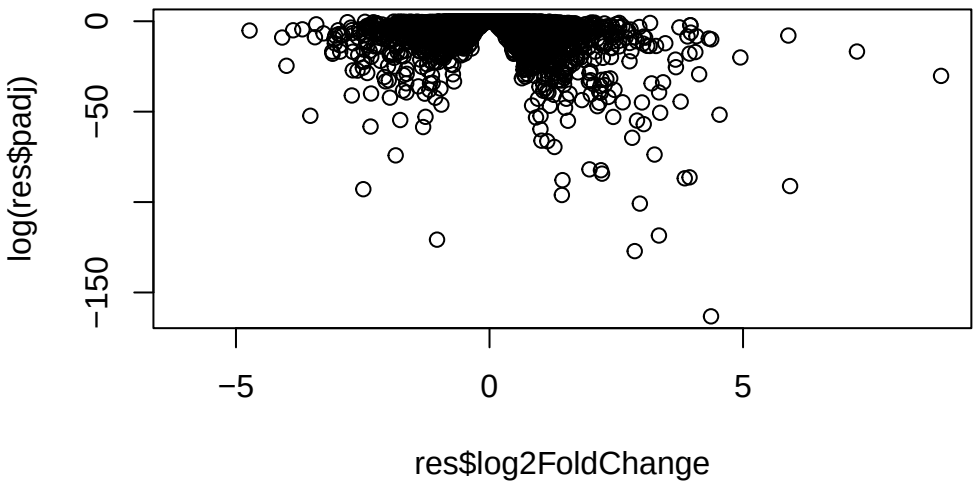
Volcano plot = common visualization for this type of data, plotting log2 fold change (X-axis) against adjusted P-value (Y-axis)

```
plot(res$log2FoldChange, res$padj)
```



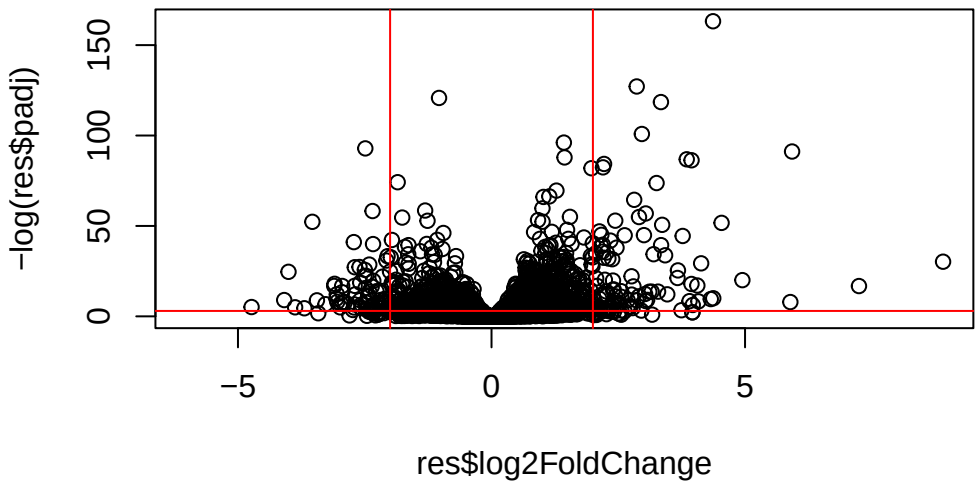
However, we actually only care about the low P-value genes. So we will log transform the data now.

```
plot(res$log2FoldChange, log(res$padj))
```

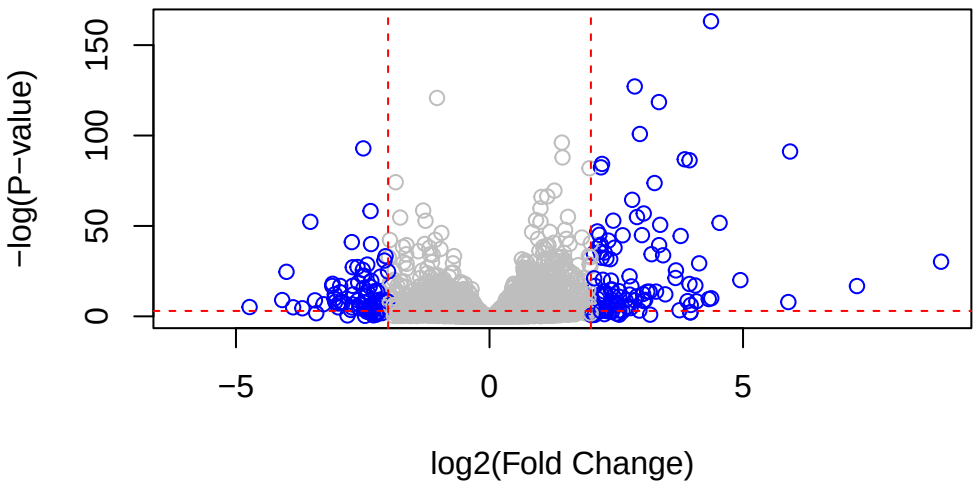


Let's flip the axis

```
plot(res$log2FoldChange, -log(res$padj))
abline(v=-2, col="red")
abline(v=2, col="red")
abline(h=-log(0.05), col="red")
```




```
mycols <- rep("gray", nrow(res))
mycols[abs(res$log2FoldChange)>2] <- "blue"
plot(res$log2FoldChange, -log(res$padj), col=mycols, ylab = "-log(P-value)", xlab = "log2(
abline(v=-2, col="red", lty=2)
abline(v=2, col="red", lty=2)
abline(h=-log(0.05), col="red", lty=2)
```



Save our results thus far.

##Creating an enhanced volcano plot. We first need to add gene names for annotation. So far, our results table only contains the Ensembl gene IDs. However, we want to be able to make sense of these genes. Minimally, we should add conventional gene symbol names, but also identifiers for other databases that we might want to look in later for information about these genes.

```
library("AnnotationDbi")
library("org.Hs.eg.db")
```

```
columns(org.Hs.eg.db)
```

```
 [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
 [6] "ENTREZID"     "ENZYME"       "EVIDENCE"     "EVIDENCEALL"  "GENENAME"
[11] "GENETYPE"     "GO"           "GOALL"        "IPI"          "MAP"
[16] "OMIM"         "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"
[21] "PMID"         "PROSITE"      "REFSEQ"       "SYMBOL"       "UCSCKG"
[26] "UNIPROT"
```

```
  res$symbol <- mapIds(org.Hs.eg.db,
                       keys=row.names(res),
                       keytype="ENSEMBL",
                       column="SYMBOL",
                       multiVals="first")
```

```
'select()' returned 1:many mapping between keys and columns
```

```
  #keys = our genenames ; keytype = our format ; column = the new format we want
```

```
  head(res$symbol)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
       "TSPAN6"          "TNMD"          "DPM1"         "SCYL3"         "FIRRM"
ENSG00000000938
          "FGR"
```

```
  res$entrez <- mapIds(org.Hs.eg.db,
                       keys=row.names(res),
                       keytype="ENSEMBL",
                       column="ENTREZID",
                       multiVals="first")
```

```
'select()' returned 1:many mapping between keys and columns
```

```
  head(res$entrez)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         "7105"          "64102"           "8813"         "57147"         "55732"
ENSG00000000938
         "2268"
```

```
  head(res)
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 8 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
                 <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
                     padj      symbol      entrez
                <numeric> <character> <character>
ENSG00000000003  0.163035      TSPAN6        7105
ENSG00000000005        NA        TNMD       64102
ENSG00000000419  0.176032        DPM1        8813
ENSG00000000457  0.961694       SCYL3       57147
ENSG00000000460  0.815849       FIRRM       55732
ENSG00000000938        NA         FGR        2268
```

Finally, let's save our results for the day as a CSV file.

```
write.csv(res, file="myresults.csv")
```