Class13: RNA Seq Analysis

Justin Lu (A16318305)

The data for today's lab comes from published RNA-seq experiment where airway smooth muscle cells were treated with deamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects.

Import Data

We need two things for this analysis: counts and metadata; these are called "countData" and "colData" in the DESeq2 world.

```
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

```
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
```

Examine Data

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

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

```
table(metadata$dex)
```

```
control treated
      4       4
```

Check on match of metadata and coldata

```
colnames(counts)
```

```
[1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"
[6] "SRR1039517" "SRR1039520" "SRR1039521"
```

```
metadata$id
```

```
[1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"
[6] "SRR1039517" "SRR1039520" "SRR1039521"
```

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

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

If you want to know that all the elements of a vector are TRUE we can use the `all()` function.

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

```
[1] TRUE
```

Analysis

I want to start by comparing "control" and "treated" columns. To do this, I can calculate the mean counts for each gene (row) in all "control" columns. Then I can calculate the mean counts for each gene in all "treated" columns and compare the two. Higher counts will imply higher gene expression.

Let's extract all "control" columns first.

```
control.inds <- metadata$dex == "control"
```

```
control.counts<- counts[,control.inds]
```

Now find the mean count value per gene using the `apply()` function.

```
control.mean <- apply(control.counts,1,mean)
```

Now I will do the same thing for "treated column"

```
treated.inds <- metadata$dex == "treated"
```

```
treated.counts<- counts[, treated.inds]
```

```
treated.mean <- apply(treated.counts,1,mean)
```

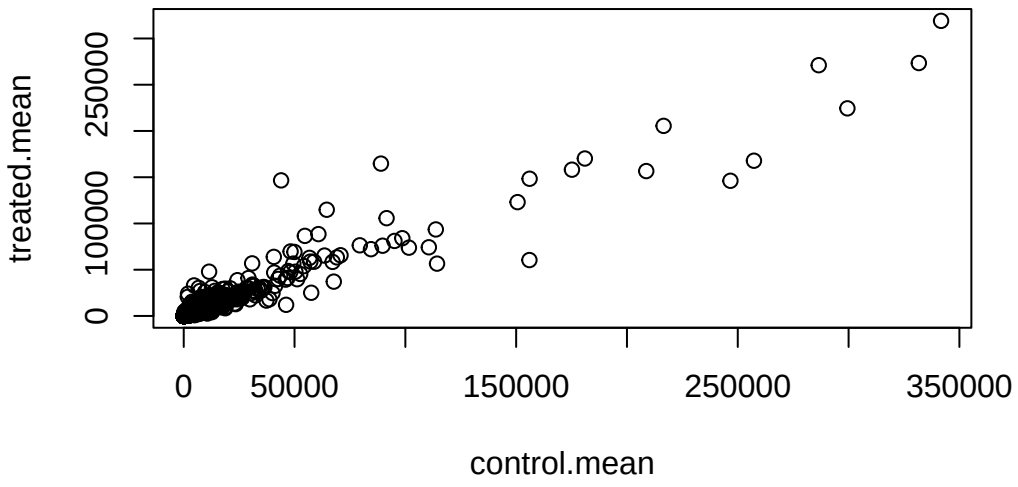
Put these two mean vectors together for ease of book-keeping.

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

```
                control.mean treated.mean
ENSG00000000003       900.75       658.00
ENSG00000000005         0.00         0.00
ENSG00000000419       520.50       546.00
ENSG00000000457       339.75       316.50
ENSG00000000460        97.25        78.75
ENSG00000000938         0.75         0.00
```

Let's have a wee look with a quick plot.

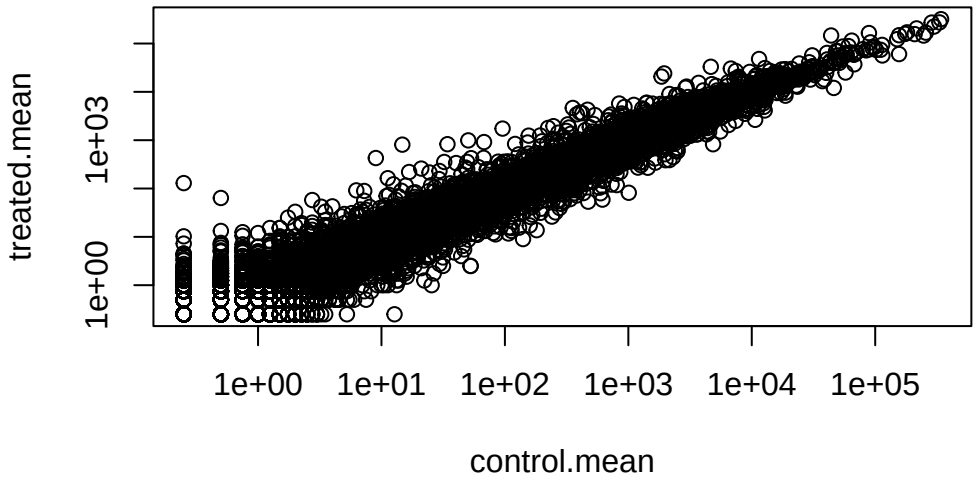
```
plot(meancounts)
```




```
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
```




```
log(10, base=2)
```

```
[1] 3.321928
```

```
log2(20/10)
```

```
[1] 1
```

We most often work in log2 units because they have a more intuitive interpretation.

Here we calculated the log2 Fold-change of treated/control values and add it to our data frame of results.

```
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
```

There are some funky answers in here like NaN (not a number) and - Inf (Infinity) that all come because I have zero count genes in my dataset.

It is common practice to filter these zero count genes out before we go too deep.

```
to.keep.inds <- (rowSums(meancounts[,1:2] == 0) == 0)

mycounts <- meancounts[to.keep.inds,]
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
```

Q. How many genes do we have left after zero count filtering/

```
nrow(mycounts)
```

```
[1] 21817
```

A common threshold for calling a gene “up” or “down” is a log2 fold change of +2 or 02.

Q. How many “up” regulated genes do we have

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

```
[1] 314
```

DESeq analysis

We need to do this analysis properly with our inner stats person kept happy.

```
library(DESeq2)
```

To use DESeq2 we need to get our input data in a very particular format.

```
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
```

Run DESeq analysis

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

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

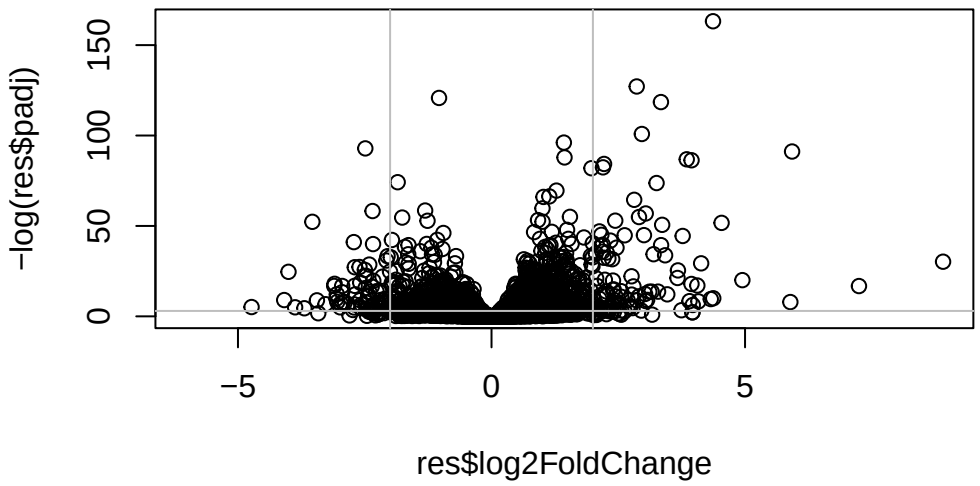
Get the results

```
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
```

I want to make a figure showing an overview of all my results to date. A plot of ***log2 fold change** vs the **p-value** (adjusted p-value)

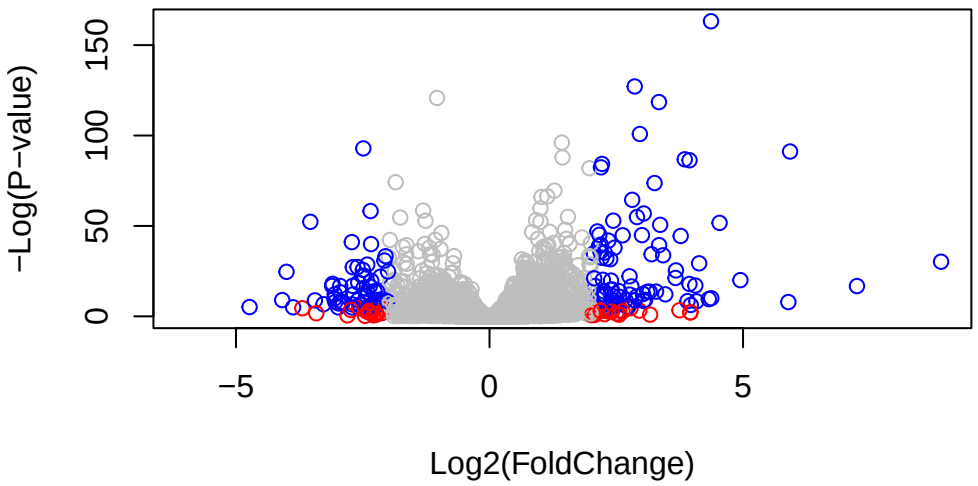
```
plot(res$log2FoldChange, -log(res$padj))
abline(v=-2, col = "gray")
abline(v=2, col = "gray")
abline(h=-log(0.05), col = "gray")
```




```
# Setup our custom point color vector
mycols <- rep("gray", nrow(res))
mycols[abs(res$log2FoldChange) > 2 ]  <- "red"

inds <- (res$padj < 0.01) & (abs(res$log2FoldChange) > 2 )
mycols[inds] <- "blue"

# Volcano plot with custom colors
plot(res$log2FoldChange,  -log(res$padj),
 col=mycols, ylab="-Log(P-value)", xlab="Log2(FoldChange)" )
```


−Log(P−value)
0
50
100
150
−5
0
5
Log2(FoldChange)