

Class 12

Raidah Anisah Huda

Here we will use the DESeq2 package for RNASeq analysis. The data for today's class came from a study of airway smooth muscle cells treated with dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al. 2014)

Import their data

We need two things for this analysis:

- **countData** (counts for every transcript/gene in each experiment)
- **colData** (metadata that describes the experimental setup)

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

	SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516
ENSG000000000003	723	486	904	445	1170
ENSG000000000005	0	0	0	0	0
ENSG000000000419	467	523	616	371	582
ENSG000000000457	347	258	364	237	318
ENSG000000000460	96	81	73	66	118
ENSG000000000938	0	0	1	0	2
	SRR1039517	SRR1039520	SRR1039521		
ENSG000000000003	1097	806	604		
ENSG000000000005	0	0	0		
ENSG000000000419	781	417	509		
ENSG000000000457	447	330	324		
ENSG000000000460	94	102	74		
ENSG000000000938	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(countData)
```

```
[1] 38694
```

Q2. How many ‘control’ cell lines do we have?

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

```
control treated
4          4
```

```
#another option
sum(metadata$dex == "control")
```

```
[1] 4
```

Toy differential gene expression

- Step 1. Calculate the mean of the control samples (i.e. columns in `countData`) Calculate the mean of the treated samples

(a) We need to find which columns in `countData` are “control” samples.

- look in the metadata (a.k.a `colData`), `$dex` column

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

(b) Extract all the control columns from `countData` and call it `control.counts`

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

- (c) Calculate the mean value across the rows of `control.counts` i.e. calculate the mean count values for each gene in the control samples.

```
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?

It is better to use `rowMeans` instead of `rowSums` because more samples were added then using `rowSums/4` wouldn't take into account the new values.

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`)

- Step 2. Calculate the mean of the treated samples...

```
treated inds <- metadata$dex=="treated"
treated counts <- countData[ , treated inds]
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
```

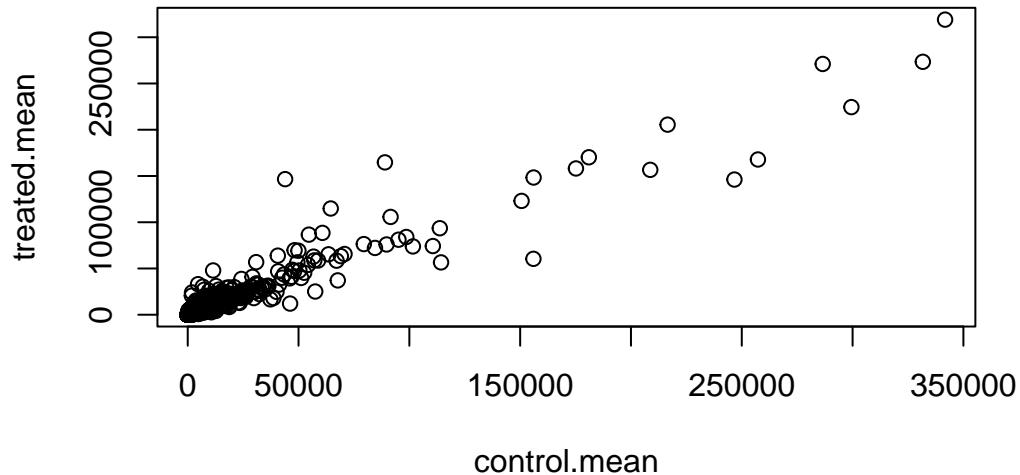
We now have control ad treated mean count values. For ease of book-keeping I will combine these vectors into a new data.frame called `meancounts`

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

	control.mean	treated.mean
ENSG000000000003	900.75	658.00
ENSG000000000005	0.00	0.00
ENSG00000000419	520.50	546.00
ENSG00000000457	339.75	316.50
ENSG00000000460	97.25	78.75
ENSG00000000938	0.75	0.00

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

```
plot(meancounts)
```



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?

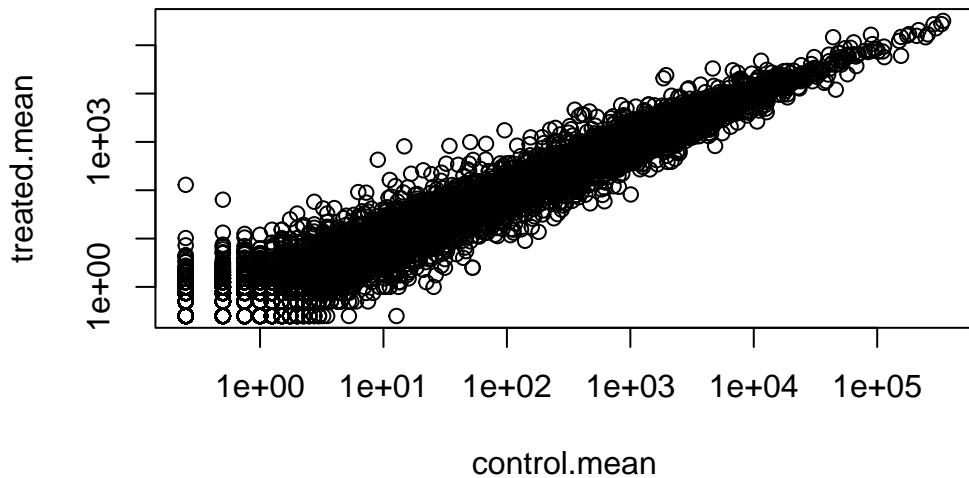
I would use geom_point for this plot.

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

```
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
```



We use log transforms for skewed data such as this and because we really care most about relative changes in magnitude.

We most often use log2 as our transfer as the math is easier to interpret than log10 or others.

If we have no change - i.e. some values in control and treated we will have a log2 value of zero.

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

```
[1] 0
```

If I have double the amount (20 compared to 10 for example) I will have a log2 fold change of +1

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

```
[1] 1
```

If I have half the amount I will have a log2 fold-change of -1

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

```
[1] -1
```

```
log2(40/10)
```

```
[1] 2
```

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

	control.mean	treated.mean	log2fc
ENSG000000000003	900.75	658.00	-0.45303916
ENSG000000000005	0.00	0.00	NaN
ENSG000000000419	520.50	546.00	0.06900279
ENSG000000000457	339.75	316.50	-0.10226805
ENSG000000000460	97.25	78.75	-0.30441833
ENSG000000000938	0.75	0.00	-Inf

Q7. Did not answer (as told by Dr. Grant) Q8. How many genes are upregulated at the common threshold of +2 log2FC values?

```
sum(meancounts$log2fc >= 2, na.rm=TRUE)
```

```
[1] 1910
```

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

```
sum(meancounts$log2fc <= 2, na.rm=TRUE)
```

```
[1] 23412
```

Hold on what about the stats!? Yes these are big changes but are these changes significant?!

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

No, I do not trust these results just yet because we havne't checked if the results are statistically significant.

To do this properly we will turn to the DESeq2 package

DESeq2 analysis

```
library(DESeq2)
```

To use DESeq we need out input countData and colDat in a specific format that DESeq wants:

```
dds <- DESeqDataSetFromMatrix(countData = countData,
                                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
```

```
class: DESeqDataSet
dim: 38694 8
metadata(1): version
assays(1): counts
rownames(38694): ENSG00000000003 ENSG00000000005 ... ENSG00000283120
ENSG00000283123
rowData names(0):
colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
colData names(4): id dex celltype geo_id
```

To run the analysis I can now use the main DESeq2 function called `DESeq()` with `dds` as input.

```

dds <- DESeq(dds)

estimating size factors

estimating dispersions

gene-wise dispersion estimates

mean-dispersion relationship

final dispersion estimates

fitting model and testing

```

To get the results out of this `dds` object we can use the `results()` function from the package.

```

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

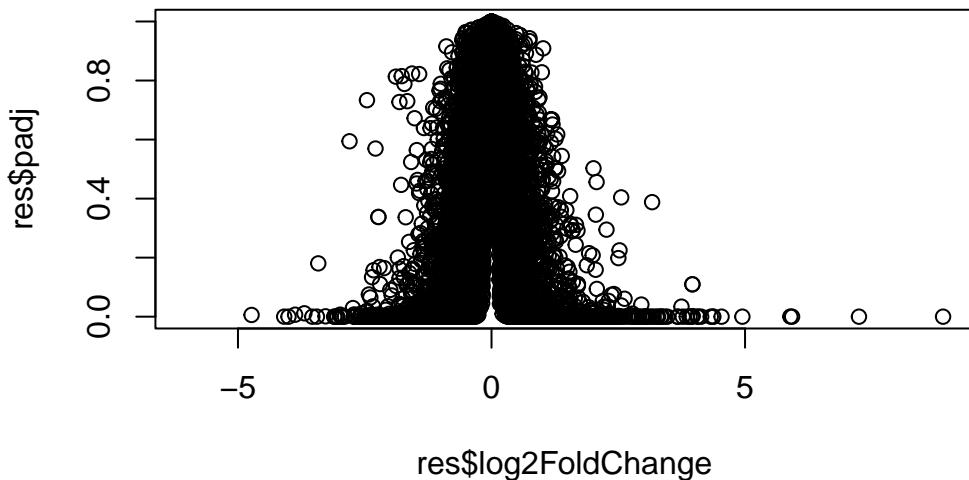
```

```
summary(res)
```

```
out of 25258 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)      : 1563, 6.2%
LFC < 0 (down)    : 1188, 4.7%
outliers [1]       : 142, 0.56%
low counts [2]     : 9971, 39%
(mean count < 10)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

Lets make a final (for today) plot of log2 fold-change vs the adjusted P-value

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

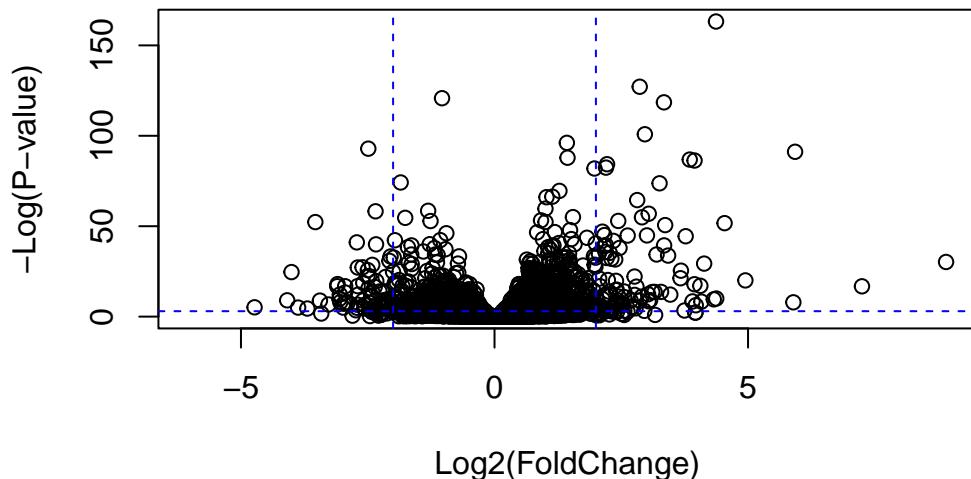


It is the low P-values that we care about and these are lost in the skewed plot above. Let's take the log of the \$padj values for our plot.

```

plot(res$log2FoldChange, -log(res$padj),
     xlab="Log2(FoldChange)",
     ylab="-Log(P-value)")
abline(v=c(-2,+2), col="blue", lty=2)
abline(h=-log(0.05), col="blue", lty=2)

```



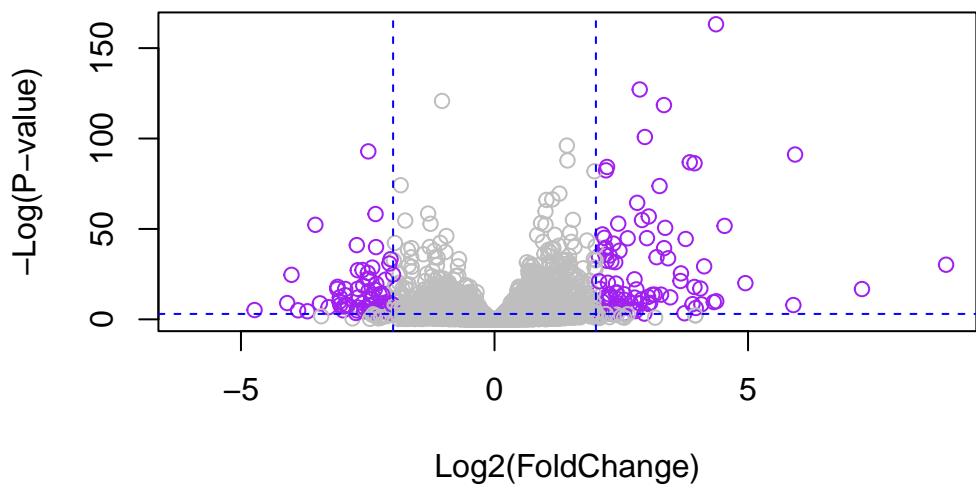
Finally we can make a color vector to use in the plot to better highlight the genes we care about.

```

mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) >= 2 ] <- "purple"
mycols[res$padj >0.05] <- "gray"

plot(res$log2FoldChange, -log(res$padj),
     xlab="Log2(FoldChange)",
     ylab="-Log(P-value)", col=mycols)
abline(v=c(-2,+2), col="blue", lty=2)
abline(h=-log(0.05), col="blue", lty=2)

```



Done for today :-)