Class 12: RNAseq Analysis

Gen Dantay (BIMM 143)

Here we will use the DESeq2 package for RNAseq analysis. The data for today's class come from a study of airway smmoth muscle cells treated with dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects.

Import CountData and ColData

We need two things for this analysis: - countData (counts for every trancript/gene in each experiment) - ColData (metadata that describes the experimental system)

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

```
                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(countData)
```

```
[1] 38694
```

There are 38,694 genes in the dataset.

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

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

```
control treated
      4       4
```

We have 4 'control' cell lines in the data. Another way:

```
sum(metadata$dex == "control")
```

```
[1] 4
```

Toy Differential Gene Expression

First, we will take the mean of just the control variables. Before doing so, we must get all the counts. - 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 are "control" samples.

- We need to look in the metadata (a.k.a colData), specifically in the $dex column

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

```
head(countData[, control.inds])
```

```
                SRR1039508 SRR1039512 SRR1039516 SRR1039520
ENSG00000000003        723        904       1170        806
ENSG00000000005          0          0          0          0
ENSG00000000419        467        616        582        417
ENSG00000000457        347        364        318        330
ENSG00000000460         96         73        118        102
ENSG00000000938          0          1          2          0
```

(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. calcualte the mean count values for each gene in the control samples.

```
control.means<- rowMeans(control.counts)
head(control.means)
```

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

- step 2: calculate the mean of the treated side: > 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"
head(countData[, treated.inds])
```

```
                SRR1039509 SRR1039513 SRR1039517 SRR1039521
ENSG00000000003        486        445       1097        604
ENSG00000000005          0          0          0          0
ENSG00000000419        523        371        781        509
ENSG00000000457        258        237        447        324
ENSG00000000460         81         66         94         74
ENSG00000000938          0          0          0          0
```

```
treated.counts<- countData[, treated.inds]
treated.means <- rowMeans(treated.counts)
head(treated.means)
```

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

We now have control and 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.means, treated.means)
head(meancounts)
```

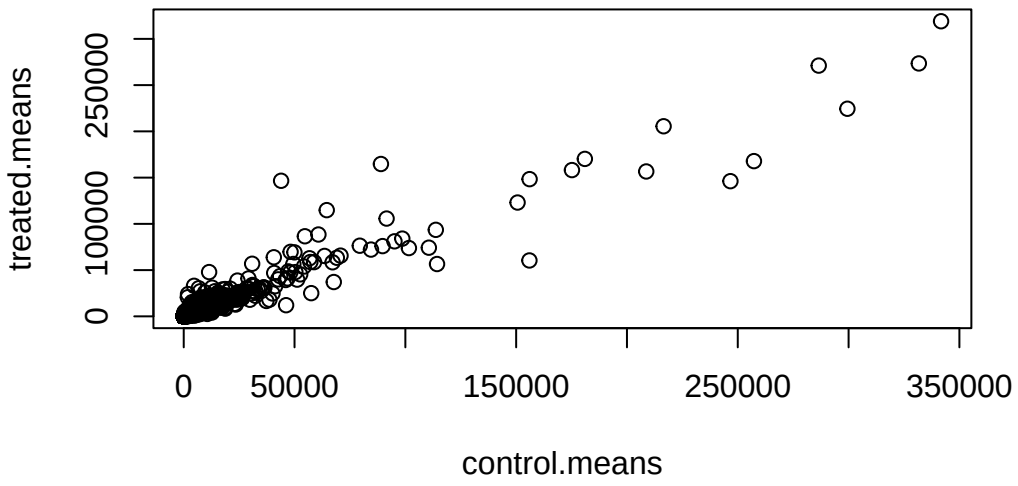
```
                control.means treated.means
ENSG00000000003        900.75        658.00
ENSG00000000005          0.00          0.00
ENSG00000000419        520.50        546.00
ENSG00000000457        339.75        316.50
ENSG00000000460         97.25         78.75
ENSG00000000938          0.75   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

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

Based on the code from the lab worksheet, we have done it in a more robust and easier way as a class, all above !

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.

```
plot(meancounts)
```

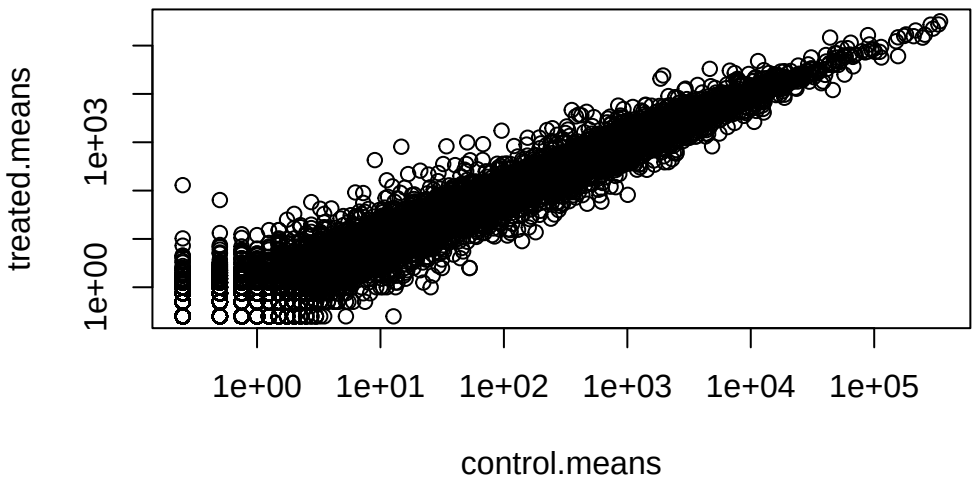


Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

```
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 used log transforms for skewed data such as this and because we really care most about relative changes in magnitude. We must often use log2 as our transform as the math is easier to interpret than log10 or others. If we have no change - i.e. same values in control and treated will have a log2 value of zero.

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

```
[1] 0
```

If I have double the amount i.e. 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.means / meancounts$control.means)
head(meancounts)
```

```
                control.means treated.means      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
```

did not do Q7... >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
```

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

No, we do not trust these results, since we do not know and have not gone over the statistics to find out whether or not the changes are statistically significant.

Hold on, what about the statistics? Yes, these are big changes but are these changes significant?

To do this properly, we will turn to DESeq2 package.

DESeq2 analysis

```
library(DESeq2)
```

To use DESeq2 we need our input countData and colData in a specific format that DESeq2 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
```

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
```

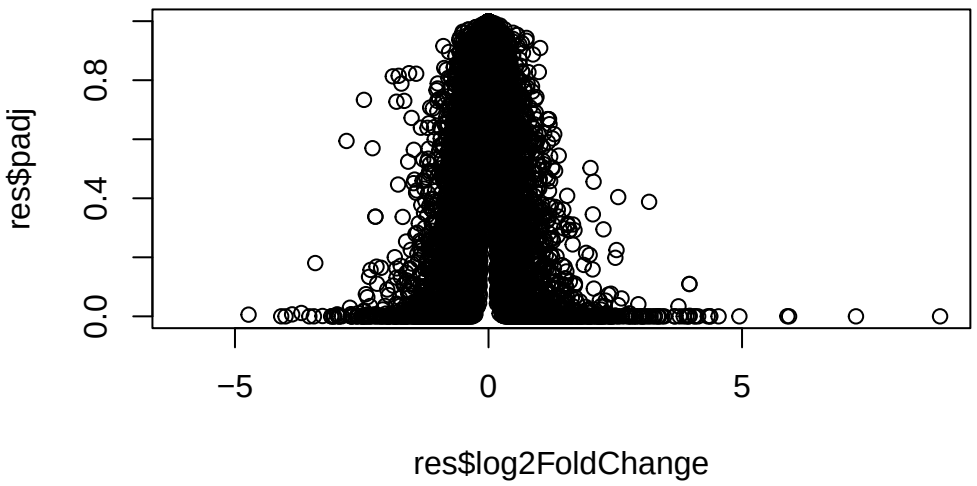
To get the results out of the `dds` object we can use the `results()` function from the package.

```
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
```

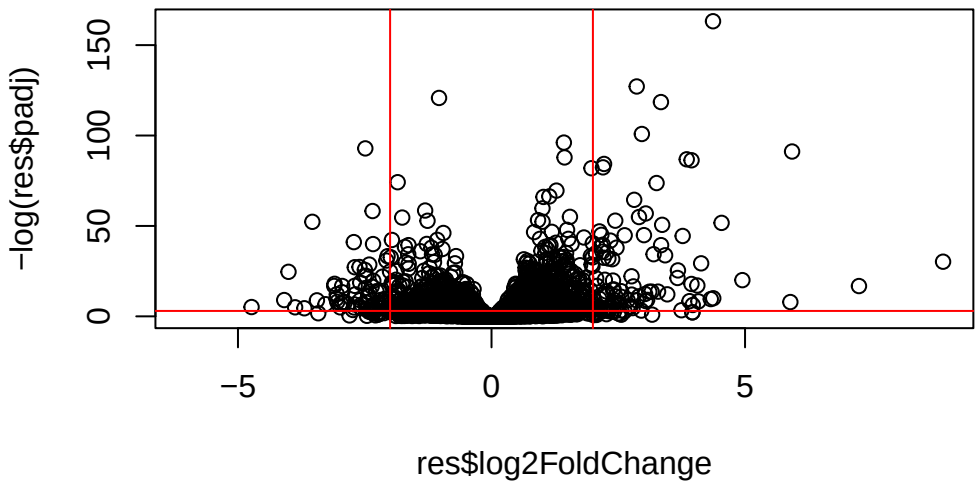
Let's make a final plot of the log2 fold change versus the adjusted p value.

```
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))
abline(v=c(+2,-2), col="red")
abline(h=-log(0.05), col= "red")
```

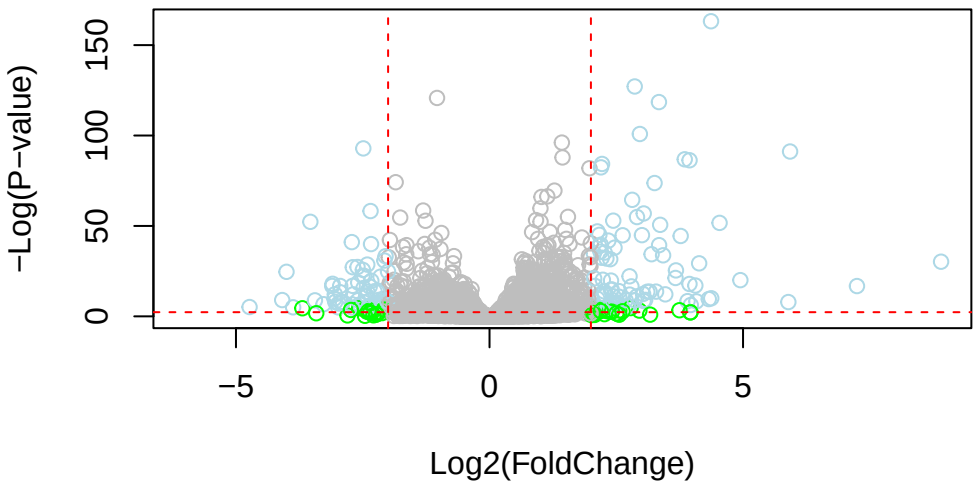



```
# Setup our custom point color vector
mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ]  <- "green"

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

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

# cut-off lines
abline(v=c(-2,2), col="red", lty=2)
abline(h=-log(0.1), col="red", lty=2)
```


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