Class 13 Transcriptomics A16246401

Scott MacLeod

Transcriptomics

In today's class we will expplore and analyze data from an RNASeq experiment where airway smooth muscles cells were treated with dexamethansone, a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al. 2014).

Data Import

We have two input files, so-called "Count data" and "col data".

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

```
              dex celltype     geo_id
SRR1039508 control   N61311 GSM1275862
SRR1039509 treated   N61311 GSM1275863
SRR1039512 control  N052611 GSM1275866
SRR1039513 treated  N052611 GSM1275867
SRR1039516 control  N080611 GSM1275870
SRR1039517 treated  N080611 GSM1275871
```

```
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
```

Q1. How many genes are in the data set?

```
nrow(counts)
```

```
[1] 38694
```

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

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

```
[1] 4
```

4. Toy differential gene expression

Time to do some analysis.

We have 4 control and 4 treated samples/experiments/columns.

Make sure the metadata ID column matches the column in our count data.

```
colnames(counts)
```

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

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

```
[1] TRUE
```

To start I will calculate the `control.mean` and `treat.mean` values and compare them.

-Identify and extract the `control` only columns. -Determine the mean value for each gene (i.e row) -Do the same for `treated`.

First the `control`.

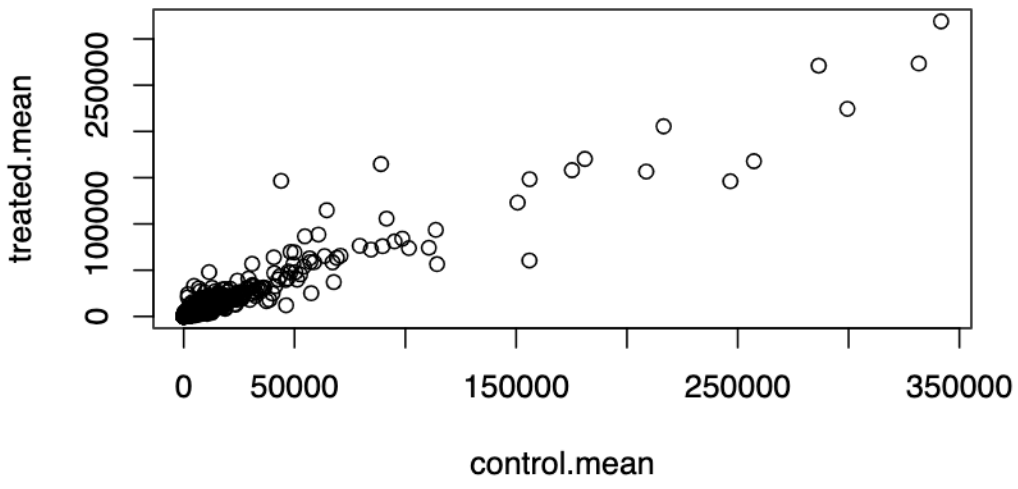
```
#Where does it tell me whicih columns are control?
control.inds <- metadata$dex == "control"
#Above shows how to extract which is control or treated. True is control
control.counts <- counts[,control.inds]
control.mean <- apply(control.counts, 1, mean)
```

Now let's get the mean for `treated`.

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

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

```
plot(meancounts)
```

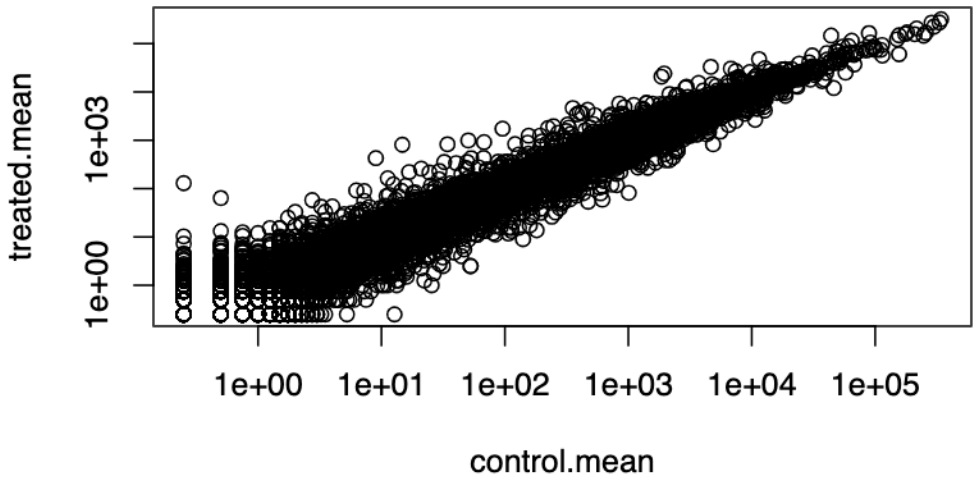


THis data is screaming at us to log transform as it is so heavily skewed and over such a wide rangee.

```
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
```



I want to compare the treated and then control values here and we will use fold change in log2 units to do this. Essentially: log2(treated/control)

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

Some log review: No difference

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

```
[1] 0
```

A doulbing in the treated:

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

```
[1] 1
```

```
log2(5/10)
```

```
[1] -1
```

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

```
[1] 2
```

```
log2(2.5/10)
```

```
[1] -2
```

A common rule of thumb cut-off for calling a gene "differentially expressed" is a log2 fold-change value of either $> +2$ or < -2 for "up regulated" and "down regulated" respectively.

```
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 first need to remove zero count genes - as we can't say anything about these genes anyway and their division of log values are messing things up (divide by zero or log of 0) or the infiinity log problem.

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

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

This table is much better!

```
to.rm.ind <- rowSums(meancounts[,1:2]==0) >0
mycounts <- meancounts[!to.rm.ind,]
```

Q. How many genes do we have left that we can say something about (i.e. they don't have any zero counts)?

```
nrow(mycounts)
```

```
[1] 21817
```

Q8. Using the up.ind vector above can you determine how many up regulated genes we have at the greater than 2 fc level?

```
up.ind <-sum(mycounts$log2fc > +2, na.rm =TRUE)
up.ind
```

```
[1] 250
```

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

```
down.ind <- sum(mycounts$log2fc < -2, na.rm =TRUE)
down.ind
```

```
[1] 367
```

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

We do trust these results becaue we have no done substanstial statistical analysis. The change is not significant! We are missing stats!

DESeq Analysis

Let's do this properly with the help of the DESeq2 package

```
library(DESeq2)
```

We have to use a specific data object for working DESeq.

```
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 our main analysis with the `DESeq()` fucntion.

```
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 our `dds` object we can use our DESeq function called `results()`:

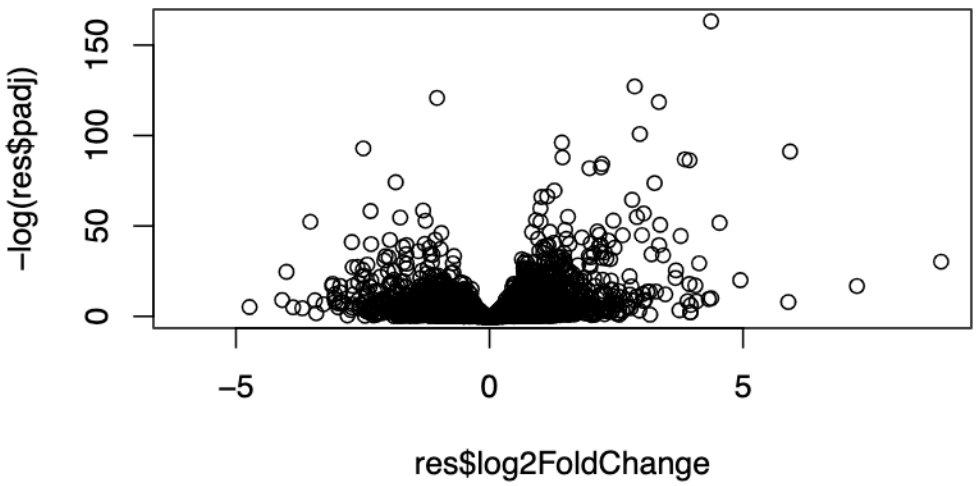
```
res <- results(dds)
head(res)
```

```
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
```

Volcano Plot

A very common and useful summary results figure from this type of analysis is called a volcano plot - a plot of log2FC vs P-value. We use the `padj`, the adjusted P-Value for multiple testing.

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



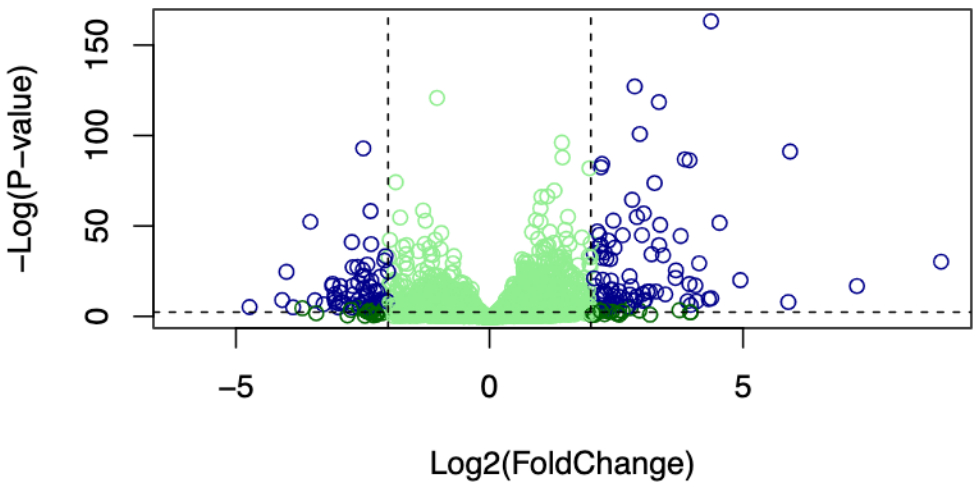
now let's add some color.

```
mycols <- rep("lightgreen", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ]  <- "darkgreen"

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

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

abline(v=c(-2,2), col="black", lty=2)
abline(h=-log(0.1), col="black", lty=2)
```




```
log(0.000005)
```

```
[1] -12.20607
```

```
log(0.05)
```

```
[1] -2.995732
```