Class 12: RNAseq

Emily Chase (PID: A14656894)

Table of contents

Background

Today we will analyze some RNASeq data from Himes et al. on the effects of a common steroid (dexamethasone ("dex")) on airway smooth muscle cells (ASMs).

For this analysis we need two main inputs

- `countData`: a table of **counts** for genes (in rows) across conditions/experiments (in columns)
- `colData`: a table of **metadata** about the design of the experiments. The rows match the columns in `countData`

DESeq2

```
# BiocManager::install("DESeq2")
library(DESeq2)
```

Data Import

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

Let's have a wee peak at our `counts` data

```
head(counts) # nt: second row is all 0s so we would typically filter that out
```

```
                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
```

Q1. How many "genes" are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

Q2. How many experiments are there?

```
# can use columns in counts OR rows in metadata
nrow(metadata)
```

```
[1] 8
```

Q3. How many "control" experiments are there?

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

```
[1] 4
```

Toy analysis

1. Extract the "control" columns from `counts`
2. Calculate the mean value for each gene in these control columns 3-4. Do the same for the "treated" columns
3. Compare these mean values for each gene

- Take advantage of how each row is a different gene

Step 1.

```
metadata$dex == "control" ## use this logical to get out the columns of counts that are cont
```

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

```
control.inds <- metadata$dex == "control"
control.counts <- counts[, control.inds]
```

```
head(control.counts)
```

```
                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
```

Step 2.

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

Step 3.

```
treated.inds <- metadata$dex != "control"
treated.counts <- counts[,treated.inds]
head(treated.counts)
```

```
                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
```

Step 4.

```
treated.means <- rowMeans(treated.counts)
```

For ease of book-keeping we can store these together in one dataframe called `meancounts`

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

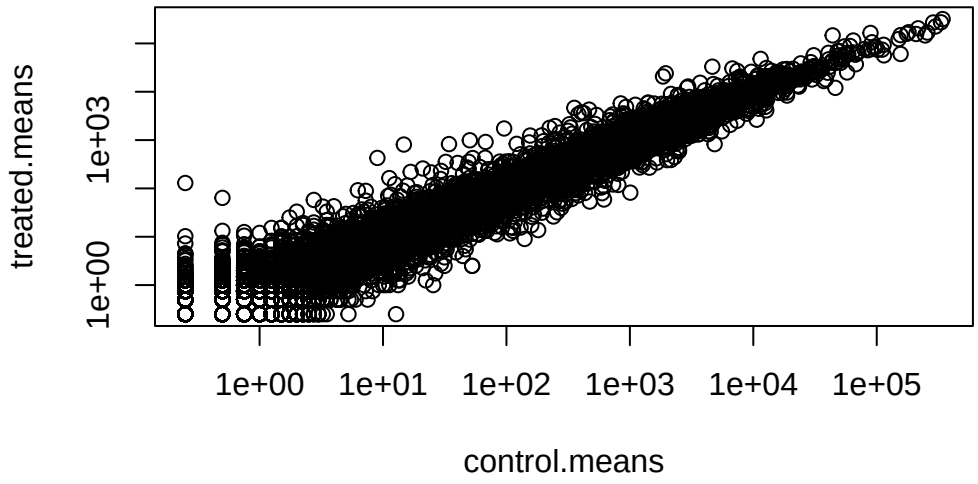
```
                control.means treated.means
ENSG00000000003        900.75        658.00
ENSG00000000005          0.00          0.00
ENSG00000000419        520.50        546.00
ENSG00000000457        339.75        316.50
ENSG00000000460         97.25         78.75
ENSG00000000938          0.75   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

Let's plot them against each other

```
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 "fold-change" as a way to compare

```
# treated/control
log2(10/10) # 0 fold change == no change
```

```
[1] 0
```

```
log2(20/10) # log2 fold change of 1
```

```
[1] 1
```

```
log2(10/20) # log2 fold change of -1
```

```
[1] -1
```

Let's add fold change as a column

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

A common "rule of thumb" for saying some thing is up/down regulated is a log-fold change >= 2 (or <=-2).

We can use pseudocounts or remove the problematic (low count) data.

```
# approach 1
# nonzero.ids <- rowSums(meancounts) != 0
# mycounts <- meancounts[nonzero.ids,]

# approach 2
zero.inds <- which(meancounts[,1:2]==0, arr.ind=T)[,1]
mygenes <- meancounts[-zero.inds,]
```

Q. How many genes are up-regulated at the +2 log2fc threshold?

```
sum(mygenes$log2fc >= 2, na.rm=T)
```

```
[1] 485
```

How many genes are down-regulated at the -2 log2fc threshold?

```
sum(mygenes$log2fc <= -2, na.rm=T)
```

```
[1] 314
```

We're unsatisfied because we don't know about significance, and we've aggregated data in a haphazard way.

DESeq analysis

Let's do this with DESeq2 and put some stats behind these numbers:

```
# library(DESeq2) # if you haven't already run it
```

DESeq wants 3 things for analysis:

- countData
- colData
- design

```
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
```

The main function in the DESeq package to run analysis is called `DESeq()`

```
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 out of this DESeq object with the function `results()`

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

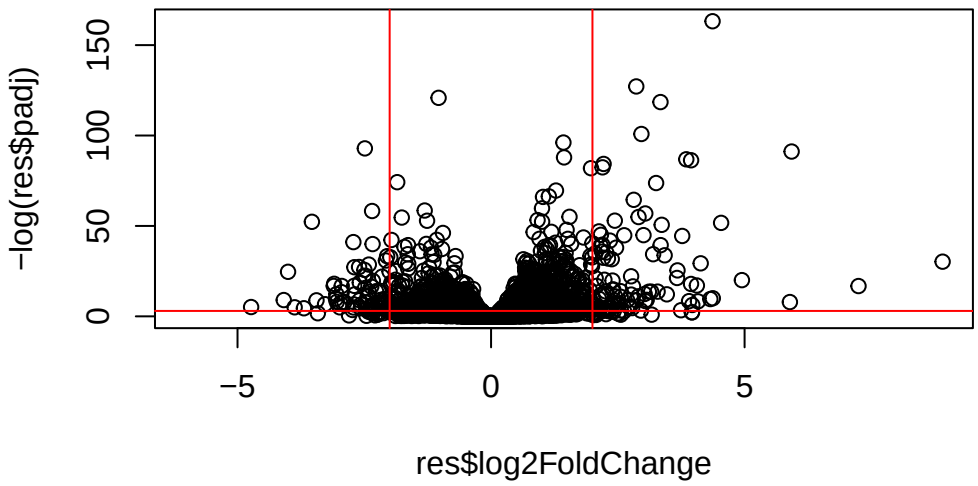
```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 6 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
                 <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195      -0.350703  0.168242 -2.084514 0.0371134
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160       0.206107  0.101042  2.039828 0.0413675
ENSG00000000457 322.664844       0.024527  0.145134  0.168996 0.8658000
ENSG00000000460  87.682625      -0.147143  0.256995 -0.572550 0.5669497
ENSG00000000938   0.319167      -1.732289  3.493601 -0.495846 0.6200029
                     padj
                <numeric>
ENSG00000000003  0.163017
ENSG00000000005        NA
ENSG00000000419  0.175937
ENSG00000000457  0.961682
ENSG00000000460  0.815805
ENSG00000000938        NA
```

We'll use the adjusted p value because it accounts for the sample size (we don't want 5% error because that's a LOT of error)

Volcano Plot

Puts log fold change on the x axis and adjusted (adj) p value on the y axis

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



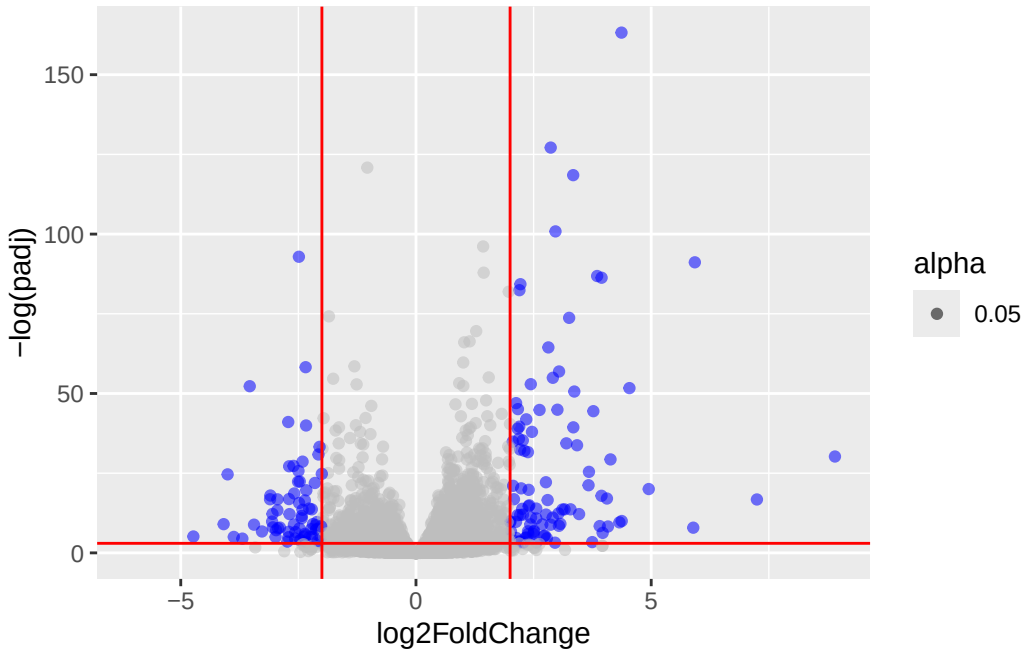
Upper left and upper right boxes are down and up regulated genes (respectively).

Color the genes that we are interested in:

```
library(ggplot2)
mycols <- rep("gray", nrow(res))
mycols[abs(res$log2FoldChange) >= 2 & res$padj<0.05] <- "blue"

ggplot(res) + aes(x=log2FoldChange, y=-log(padj), alpha = 0.05) + geom_point(col=mycols) + g
```

```
Warning: Removed 23549 rows containing missing values or values outside the scale range
(`geom_point()`).
```



Save our results

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