Class 12

Sindy Chavez

Table of contents

1. Bioconductor and DESeq2 setup

```
library(BiocManager)
library(DESeq2)
```

Input Data

We need at least two things: - count data (genes in rows and exp in cols) - metadata (a.k.a. 'colData')

2. Import countData and colData

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

Let's take a quick look at 'counts' and 'metadata'

```
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
```

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

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

```
metadata$dex
```

```
[1] "control" "treated" "control" "treated" "control" "treated" "control"
[8] "treated"
```

3. Toy differential gene expression

We need to make sure that the metadata (i.e. colData) and our counts match!

```
metadata$id
```

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

```
colnames(counts)
```

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

We can use the '==' test for equality. Basically is the bit on the right the same as the bit on the left of the == sign.

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

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

We can use the 'all()' function to check if al the inputs are TRUE.

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

```
[1] TRUE
```

Find the mean count values per gene for control samples

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

We want a mean value acress these rows (ie a mean count per gene)

```
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
```

Find the mean count values per gene for treated samples

Question 4

```
treated.inds <- metadata$dex == "treated"
treated.ids <- metadata[treated.inds,"id"]
treated.counts <- counts[,treated.ids]
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
```

```
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
```

or

you can use this chunk:'treated.mean <- rowMeans(counts[, metadata[metadata$dex == 'treated',]id])' it's just too hard to read combine control.mean and treated. means into a data.frame

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

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

In the original instructions we divided the 'rowSums' by 4, but to make the approach more robust we can divid by the number of rows, instead of 4, or we can find the means by using 'rowMeans'

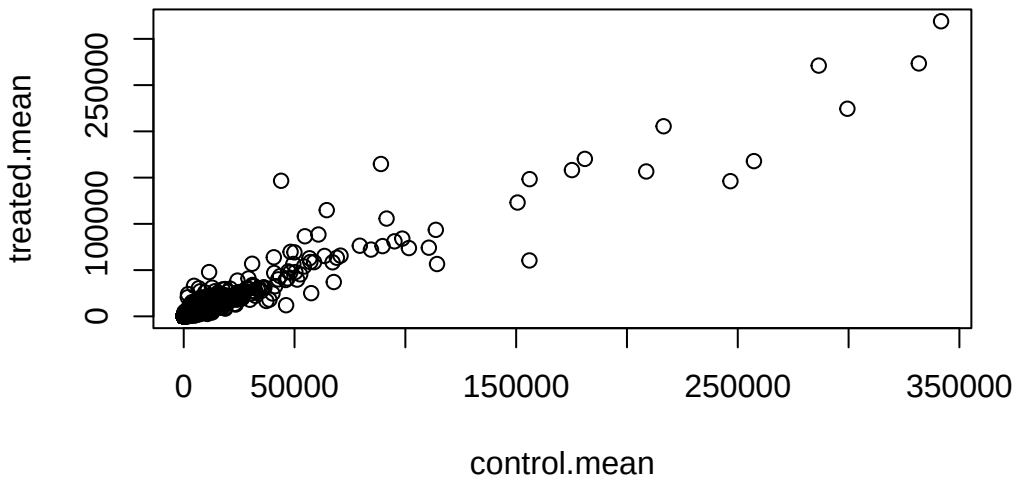
> Q4. Follow the same procedure for the treated samples (i.e. calculate the mean per gene across drug treated samples and assign to a labeled vector called treated.mean)

See above

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

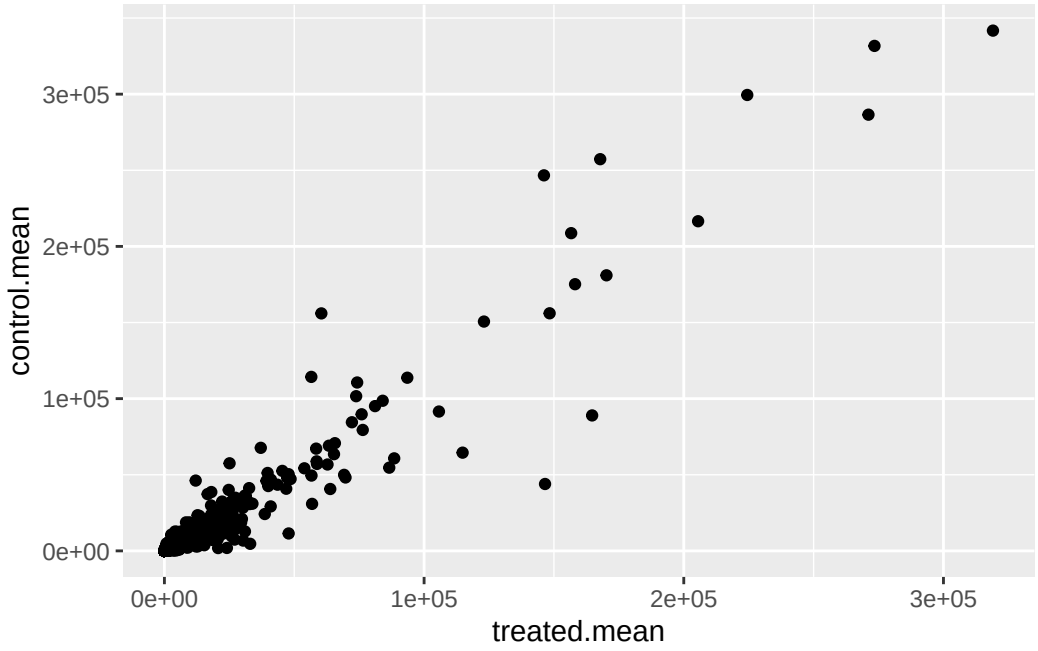
Quick plot (Q5a)

```
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?

```
library(ggplot2)
ggplot(meancounts) +
  aes(treated.mean, control.mean) +
  geom_point()
```



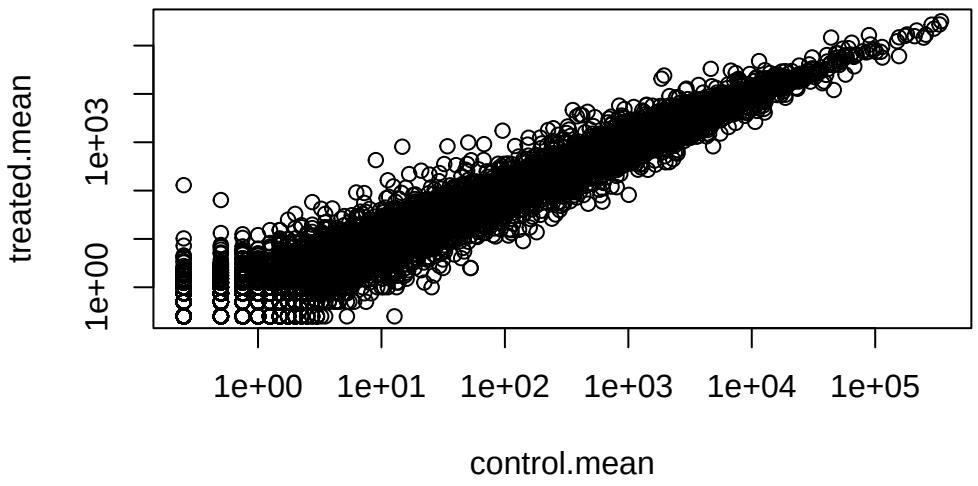
Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

This kind of heavily skewed data screams at you to log transform!

```
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
```




```
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
ENSG00000000003       900.75       658.00
ENSG00000000419       520.50       546.00
ENSG00000000457       339.75       316.50
ENSG00000000460        97.25        78.75
ENSG00000000971      5219.00      6687.50
ENSG00000001036      2327.00      1785.75
```

Q7. What is the purpose of the arr.ind argument in the which() function call above? Why would we then take the first column of the output and need to call the unique() function?

arr.ind tells us which genes have 0 values (which we can't use and want to get rid of).

I think the unique function makes sure we don't count a zero row twice if it has zeros in more thn one column.

Log2 transforms are very useful!

```
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
```

I want to get rid of any zero count genes - I can't say anythings about these genes and this drug treatment anyway.

```
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
```

How many genes do we have left?

```
nrow(mycounts)
```

```
[1] 21817
```

How many genes are "up" regulated at a threshold log2-fold-change of +2 or greater

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

```
[1] 314
```

```
up.ind <- mycounts$log2fc > 2
down.ind <- mycounts$log2fc < (-2)
sum(up.ind)
```

```
[1] 250
```

```
sum(down.ind)
```

```
[1] 367
```

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

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?

367

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

Not yet, we don't have any statistical analysis on the data to see if those changes are significant

4. DESeq2 analysis

Time to do things the way the rest of the workd do them

With DESeq2

It wants counts and colData and the "design" what to compare to what.

```
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
```

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

```
estimating size factors

estimating dispersions

gene-wise dispersion estimates

mean-dispersion relationship

final dispersion estimates

fitting model and testing
```

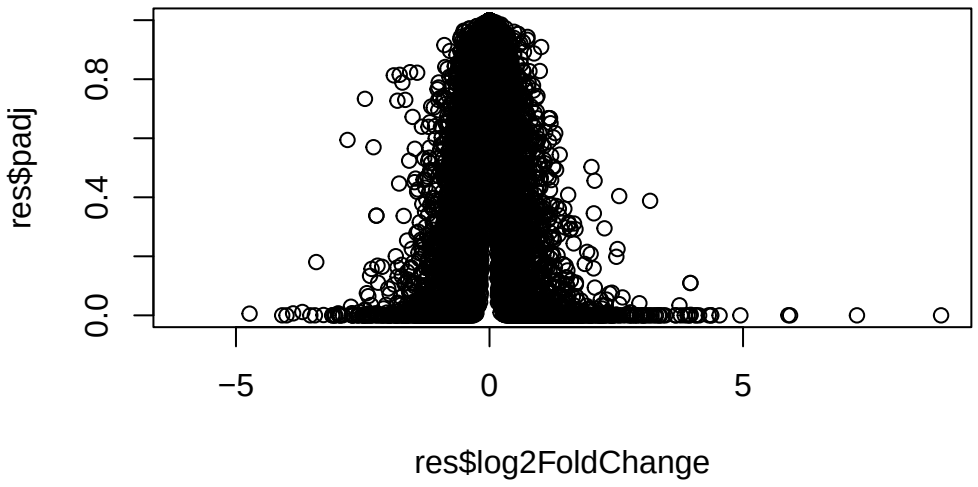
```
res <- results(dds)
```

```
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
```

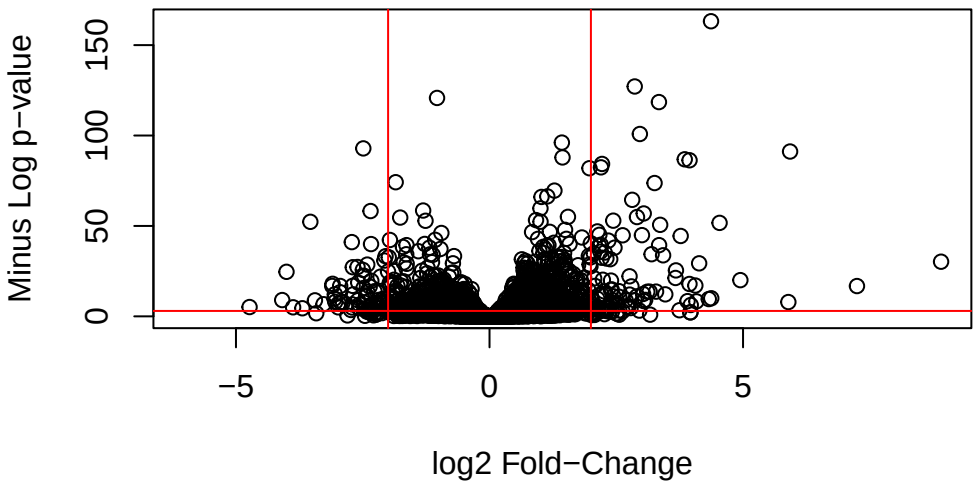
To keep both our inner biologist and inner nerd happy we often view our data in plots of log2 fold change vs p-value

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



We can take the log of the p-value to help us here again.

```
plot(res$log2FoldChange, -log(res$padj),
     xlab="log2 Fold-Change",
     ylab="Minus Log p-value")
abline(v=c(-2,2), col="red")
abline(h= -log(0.05), col="red")
```


Minus Log p−value
0
50
100
150
−5
0
5
log2 Fold−Change