Class12

Gregory Jordan

1. Bioconductor and DESeq2 Setup

```
#install.packages(BiocManager)
#library(BiocManager)
#BiocManager::Install(DESeq2)
library(DESeq2)
```

2. Import CountData and ColData

Input Data:

we need at least 2 things: -count data (genes in rows and exp in cols) -metadata (a.k.a. `colData`)

```
counts <- read.csv("https://bioboot.github.io/bimm143_W18/class-material/airway_scaledcoun
metadata <- read.csv("https://bioboot.github.io/bimm143_W18/class-material/airway_metadata
```

```
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
```

```
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
```

We need to make sure that the metadata 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"
```

```
#all function to test if every occurance is True
all(metadata$id==colnames(counts))
```

```
[1] TRUE
```

Q1. How many genes are in this dataset?

```
#goood ol' nrow
nrow(counts)
```

```
[1] 38694
```

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

```
#filtering by dex==control
nrow(metadata[metadata$dex=="control",])
```

```
[1] 4
```

3. Toy Differential Expression

First silo control and treatment into separate datasets with their corresponding count data

```
control <- metadata[metadata$dex=="control",]
head(control)
```

```
         id     dex celltype     geo_id
1 SRR1039508 control   N61311 GSM1275862
3 SRR1039512 control  N052611 GSM1275866
5 SRR1039516 control  N080611 GSM1275870
7 SRR1039520 control  N061011 GSM1275874
```

```
#link control metadata to the control ids
control.counts <- counts[,control$id]
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
```

```
#to calcute means for each gene use rowmeans
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
```

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

You could combine all of the chunks onto one line, because R does not care how it is input. However, people care! Definitely do it in multiple lines so it is easy to follow

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)

```
#filter by dex==treated
treated <- metadata[metadata$dex=="treated",]
head(treated)
```

```
         id     dex celltype     geo_id
2 SRR1039509 treated   N61311 GSM1275863
4 SRR1039513 treated  N052611 GSM1275867
6 SRR1039517 treated  N080611 GSM1275871
8 SRR1039521 treated  N061011 GSM1275875
```

```
#link treatment metadata to treatment id
treated.counts <- counts[,treated$id]
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
```

```
#get row means
head(treated.means<-rowMeans(treated.counts))
```

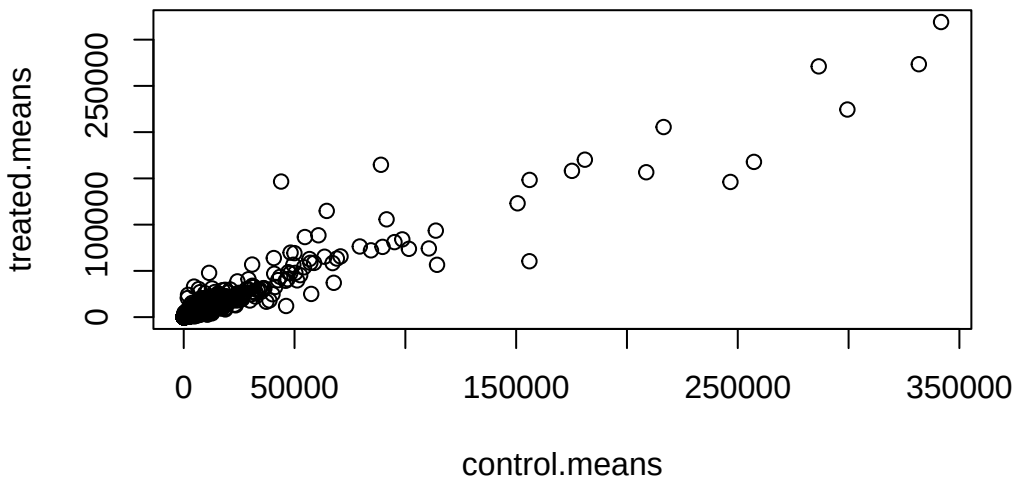
```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         658.00            0.00          546.00          316.50           78.75
ENSG00000000938
           0.00
```

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.

create a dataframe of control and treated means

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

```
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?

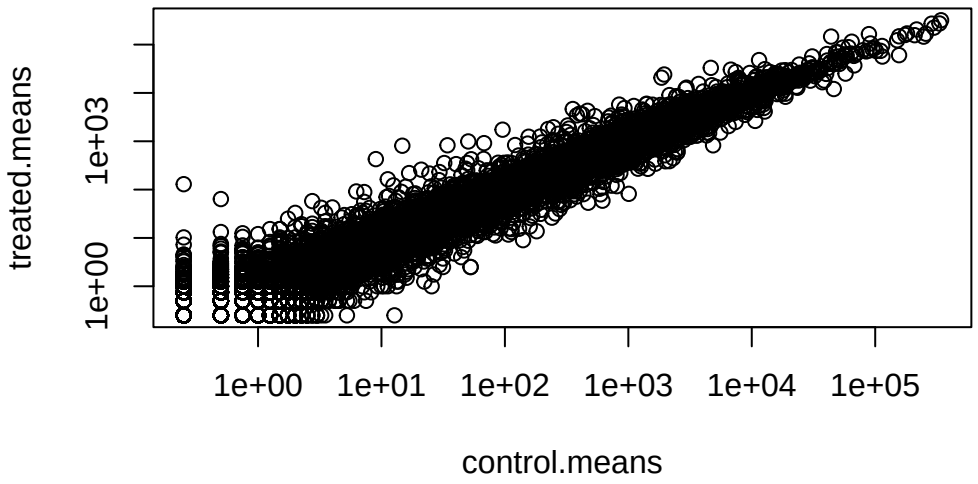
geom_point()

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

```
#use log argument to logscale your axes
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
```



log2 transforms are useful to determine fold change for genes in control vs treated. This is b/c log2 of 1 is 0, log2 of 2 is 1, log2 of 0.5 is -1, etc.

Log2 Fold Change

```
#make dataframe of meancounts
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
```

need to remove the -infinity (had zero expression) and NA values (data missing)

```
mycounts <- meancounts[rowSums(meancounts[,1:2] == 0) == 0,]
head(mycounts)
```

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

```
nrow(mycounts)
```

```
[1] 21817
```

about 20,000-25,000 genes in the human genome. this looks good

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?

The arr.ind=TRUE argument will clause which() to return both the row and column indices (i.e. positions) where there are TRUE values. In this case this will tell us which genes (rows) and samples (columns) have zero counts. We are going to ignore any genes that have zero counts in any sample so we just focus on the row answer. Calling unique() will ensure we dont count any row twice if it has zer entries in both samples.

Commonly called that Log2 fold change of +2 is considered upregulated (and vice versa for downregulated)

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

```
#we did not use the up.ind vector in class
sum(mycounts$log2fc>=2)
```

```
[1] 314
```

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

```
#we did not use the down.ind vector in class
sum(mycounts$log2fc<=-2)
```

```
[1] 485
```

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

Not really. We don't have any statistical significance values

4. DESeq2 Analysis

Time to do things the way the rest of the world does them. with DeSeq2

DeSeq2 wants counts and coldata and the "design" i.e. what to compare to what

```
#use the tilda ~ to get the design column
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
```

```
#use DESeq to do what we just did except better and with p values
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)
```

```
res
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 38694 rows and 6 columns
                 baseMean log2FoldChange     lfcSE      stat    pvalue
                <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003  747.1942     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005    0.0000             NA        NA        NA        NA
ENSG00000000419  520.1342      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457  322.6648      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460   87.6826     -0.1471420  0.257007 -0.572521 0.5669691
...                    ...            ...       ...       ...       ...
ENSG00000283115  0.000000             NA        NA        NA        NA
ENSG00000283116  0.000000             NA        NA        NA        NA
ENSG00000283119  0.000000             NA        NA        NA        NA
ENSG00000283120  0.974916      -0.668258   1.69456 -0.394354  0.693319
ENSG00000283123  0.000000             NA        NA        NA        NA
                     padj
                <numeric>
ENSG00000000003  0.163035
ENSG00000000005        NA
ENSG00000000419  0.176032
ENSG00000000457  0.961694
ENSG00000000460  0.815849
...                   ...
ENSG00000283115        NA
ENSG00000283116        NA
ENSG00000283119        NA
ENSG00000283120        NA
ENSG00000283123        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
```

```
#if we wanted we can add the alpha argument to change significance we care about
res05<-results(dds,alpha=0.5)
summary(res05)
```

```
out of 25258 with nonzero total read count
adjusted p-value < 0.5
LFC > 0 (up)       : 3375, 13%
LFC < 0 (down)     : 3108, 12%
outliers [1]       : 142, 0.56%
low counts [2]     : 8564, 34%
(mean count < 4)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

5. Adding Annotation Data

Q11. Run the mapIds() function two more times to add the Entrez ID and UniProt accession and GENENAME as new columns called res$entrez, res$uniprot and res$genename.

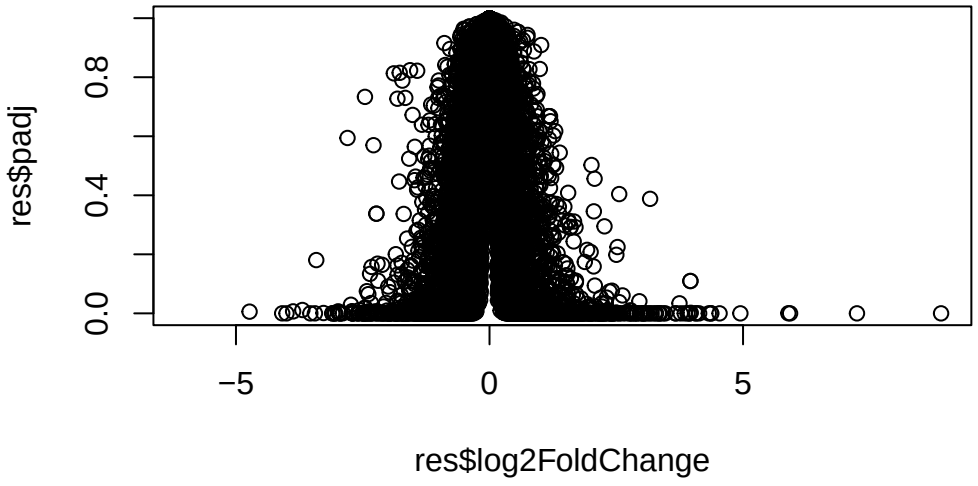
we skipped this in class

6. Data Visualization

Volcano Plots

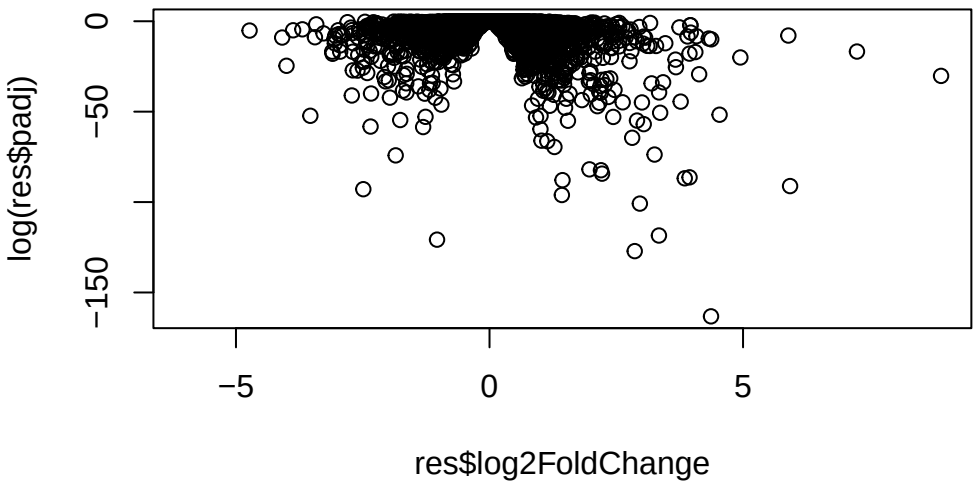
Let's see log2 up/down regulation and the significance graphed using a volcanooooo plot

```
#remember to make it log2 fold change
plot(res$log2FoldChange,res$padj)
```



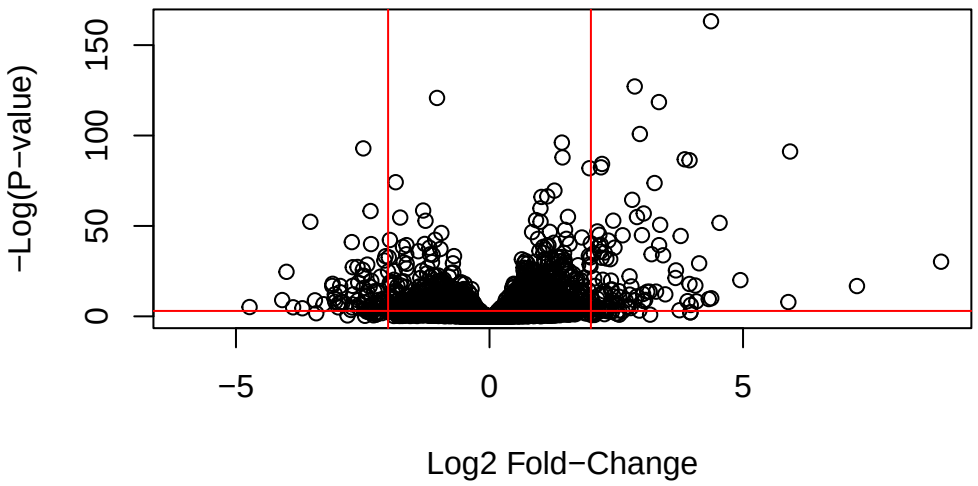
This graph is coolish, but ugly and not informative enough. let's take the log of the p value to adjust

```
#log scale now
plot(res$log2FoldChange,log(res$padj))
```



ok this is cool now but we usually want to associate lower p values with higher signfiicance so let's flip it to make it easier to interpret

```
#use abline to add lines to cut off our data based off what we want. i.e. +/-2 fold change
plot(res$log2FoldChange,-log(res$padj),xlab="Log2 Fold-Change",ylab="-Log(P-value)")
abline(v=2,col="red")
abline(v=-2,col="red")
abline(h=-log(.05),col="red")
```


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