

Class 12: RNASeq Analysis

Kaitlyn Powell

In today's class, we will work with published RNA-seq experiment where airway smooth muscle cells (ASMs) were treated with dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al. 2014).

1. Bioconductor and DESeq2 setup

```
#install.packages("BiocManager")
#BiocManager::install()

# For this class, you'll also need DESeq2:
#BiocManager::install("DESeq2")

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

2. Import countData and colData

Data Import

We will use good old `read.csv()` to read the two things we need for this analysis:

- count data
- col data (metadata)

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

How many genes do I have?

```
nrow(counts)
```

```
[1] 38694
```

Let's have a look at the 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
```

```
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
7 SRR1039520 control    N061011 GSM1275874
8 SRR1039521 treated    N061011 GSM1275875
```

And the counts data

```
head(counts)
```

	SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516
ENSG000000000003	723	486	904	445	1170
ENSG000000000005	0	0	0	0	0
ENSG000000000419	467	523	616	371	582
ENSG000000000457	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		

First we should check the correspondence of the metadata and count data.

```
metadata$id
```

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

```
colnames(counts)
```

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

To check that these are all in the same order, we can use `==` test of equality.

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

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

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

```
[1] TRUE
```

Q1. How many genes are in this dataset?

There are 38,694 genes in this dataset.

Q2. How many ‘control’ cell lines do we have?

There are 4 ‘control’ cell lines.

#3. Toy differential gene expression

Analysis via comparison of CONTROL vs TREATED

The “treated” have the dex drug and the “control” do not. First, I need to be able to extract just the “control” columns in the `counts` data set.

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

```
[1] "SRR1039508" "SRR1039512" "SRR1039516" "SRR1039520"
```

Now I can use this to access just the “control” columns of my `counts` data...

```
control.counts <- counts[,control$id]  
head(control.counts)
```

	SRR1039508	SRR1039512	SRR1039516	SRR1039520
ENSG000000000003	723	904	1170	806
ENSG000000000005	0	0	0	0
ENSG000000000419	467	616	582	417
ENSG000000000457	347	364	318	330
ENSG000000000460	96	73	118	102
ENSG000000000938	0	1	2	0

Find the mean count value for each transcript/gene by binding the `rowMeans()`.

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

ENSG000000000003	ENSG000000000005	ENSG000000000419	ENSG000000000457	ENSG000000000460
900.75	0.00	520.50	339.75	97.25
ENSG000000000938				
0.75				

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

The above code could be made more robust by simplifying the lines of code that we used in order to find the `control.mean` value. This simplified code is shown below for the treated samples:

```
treated.id <- metadata[metadata$dex == "treated", "id"]
treated.mean <- rowMeans(counts[,treated.id])
```

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)

The code for finding the mean per gene across the drug treated samples is shown below. This code is assigned to a albeled vector called `treated.mean`.

```
head(treated.mean)
```

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

And now find a mean value for all the “treated” columns in the same way.

```
treated.id <- metadata[metadata$dex == "treated", "id"]
treated.mean <- rowMeans(counts[,treated.id])
```

Now I have `control.mean` and `treated.mean`. Let's put them together for safe keeping and ease of use later.

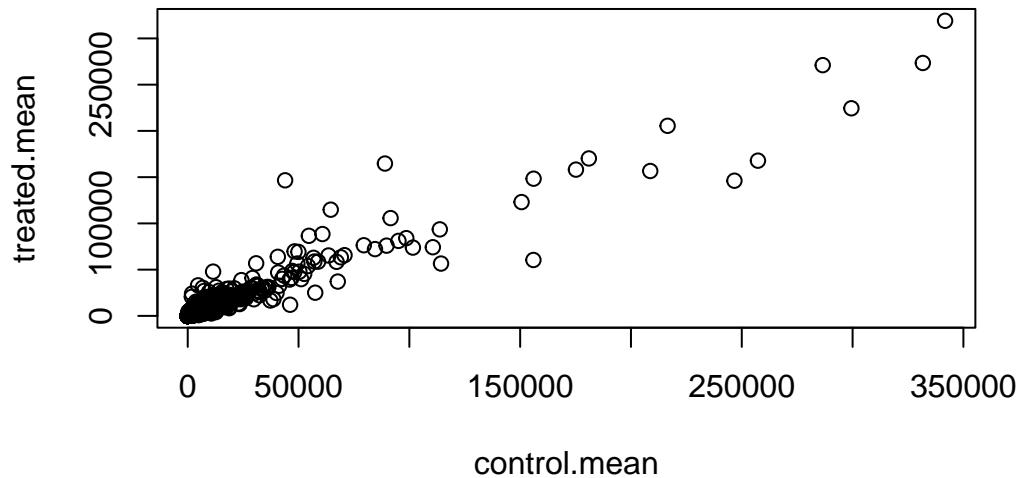
```
meancounts <- data.frame(control.mean, treated.mean)
head(meancounts)
```

	control.mean	treated.mean
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 do a quick plot to see how this looks.

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.

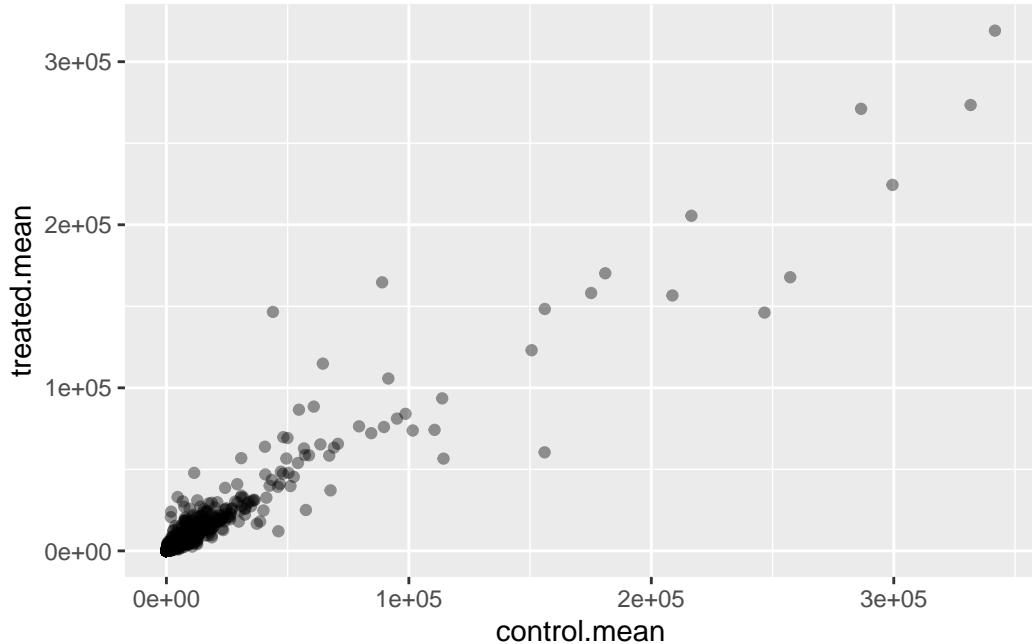
```
plot(meancounts)
```



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?

The geom_() function that would be used for this plot would be `geom_point(alpha = 0.4)`.

```
library(ggplot2)
ggplot(meancounts, aes(control.mean, treated.mean)) +
  geom_point(alpha = 0.4)
```



This is very heavily skewed and over a wide range - calling out for a log transform!

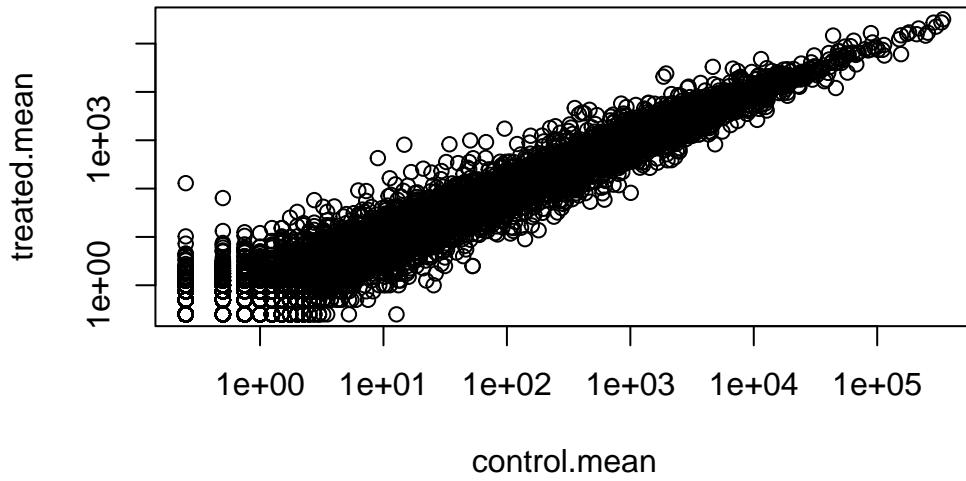
Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

The argument to plot() that allows us to plot both axes on a log scale is `plot(meancounts, log = "xy")`

```
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 like working with log transformed data as it can help make things more straightforward to interpret.

If we have no change:

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

```
[1] 0
```

What about if we had a doubling:

```
log2(40/20)
```

```
[1] 1
```

Half as much:

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

```
[1] -1
```

```
log2(80/20)
```

```
[1] 2
```

We like working with log2 fold-change values. Let's calculate them for our data.

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

	control.mean	treated.mean	log2fc
ENSG000000000003	900.75	658.00	-0.45303916
ENSG000000000005	0.00	0.00	NaN
ENSG000000000419	520.50	546.00	0.06900279
ENSG000000000457	339.75	316.50	-0.10226805
ENSG000000000460	97.25	78.75	-0.30441833
ENSG000000000938	0.75	0.00	-Inf

```
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
ENSG000000000003	900.75	658.00	-0.45303916
ENSG000000000419	520.50	546.00	0.06900279
ENSG000000000457	339.75	316.50	-0.10226805
ENSG000000000460	97.25	78.75	-0.30441833
ENSG000000000971	5219.00	6687.50	0.35769358
ENSG000000001036	2327.00	1785.75	-0.38194109

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 purpose of the arr.ind = TRUE argument in the which() function call above is to return the rows (genes) and columns (samples) that have TRUE values, which have values of zero counts. We then call the unique() function in order to ensure that we do not return the same row more than once. This can be useful if a given row has more a value of zero for both samples.

We want to filter out any genes (that is rows) where we have ZERO count data.

```
to.keep inds <- rowSums(meancounts[,1:2] == 0) == 0  
head(to.keep inds)
```

```
ENSG00000000003 ENSG00000000005 ENSG000000000419 ENSG000000000457 ENSG000000000460  
TRUE FALSE TRUE TRUE TRUE  
ENSG000000000938  
FALSE
```

```
mycounts <- meancounts[to.keep inds,]  
nrow(mycounts)
```

```
[1] 21817
```

A common threshold for calling genes as differentially expressed is a log2 fold-change of +2 or -2.

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

```
[1] 314
```

What percent is this?

```
round((sum(mycounts$log2fc >= +2) / nrow(mycounts)) * 100, 2)
```

```
[1] 1.44
```

And down regulated:

```
round((sum(mycounts$log2fc <= -2) / nrow(mycounts)) * 100, 2)
```

```
[1] 2.22
```

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

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

Using the up.ind vector, there are 250 up-regulated genes that are at the greater than 2 fc level.

```
sum(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?

Using the down.ind vector, there are 367 down-regulated genes that are at the greater than 2 fc level.

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

```
[1] 367
```

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

These results are not very trustworthy because there is no statistical data that indicates the significance. There is no p-value for these results.

We need some stats to check if the drug induced difference is significant!

Turn to DESeq2

Let's turn to doing this the correct way with the DESeq2 package.

```
library(DESeq2)
```

The main function in the DESeq2 package is called `deseq()`. It wants our count data and our colData (metadata) as input in a specific way.

```
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
```

Now I can run the DESeq analysis:

```
  dds <- DESeq(dds)  
  
  estimating size factors  
  
  estimating dispersions  
  
  gene-wise dispersion estimates  
  
  mean-dispersion relationship  
  
  final dispersion estimates  
  
  fitting model and testing  
  
  results(dds)  
  
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>  
ENSG000000000003  747.1942    -0.3507030  0.168246 -2.084470  0.0371175  
ENSG000000000005   0.0000       NA        NA        NA        NA  
ENSG000000000419  520.1342    0.2061078  0.101059  2.039475  0.0414026  
ENSG000000000457  322.6648    0.0245269  0.145145  0.168982  0.8658106  
ENSG000000000460   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>
ENSG000000000003 0.163035
ENSG000000000005 NA
ENSG000000000419 0.176032
ENSG000000000457 0.961694
ENSG000000000460 0.815849
...
...
ENSG00000283115 NA
ENSG00000283116 NA
ENSG00000283119 NA
ENSG00000283120 NA
ENSG00000283123 NA

```

Now what we have got so far is the log2 fold-change and the adjusted p-value for the significance.

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

```
head(res)
```

```

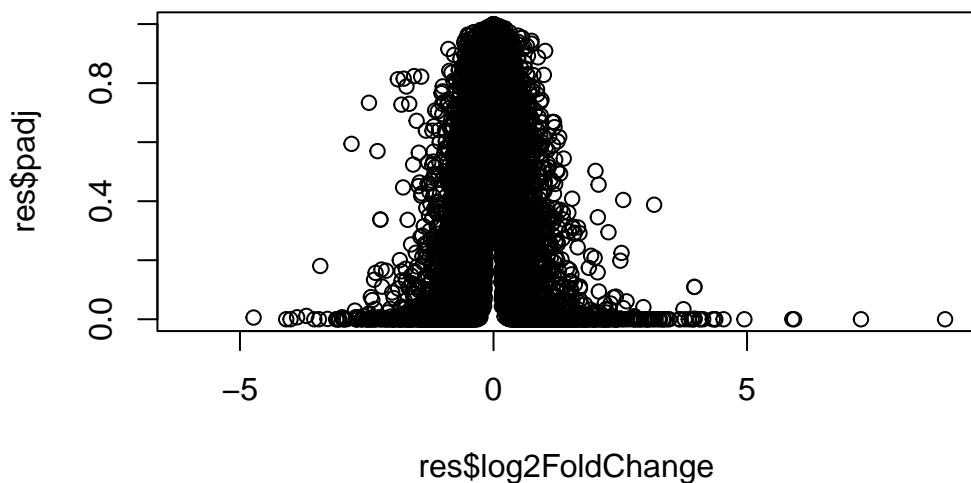
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 6 columns
      baseMean log2FoldChange     lfcSE      stat    pvalue
      <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG000000000003 747.194195 -0.3507030 0.168246 -2.084470 0.0371175
ENSG000000000005 0.000000    NA        NA        NA        NA
ENSG000000000419 520.134160 0.2061078 0.101059 2.039475 0.0414026
ENSG000000000457 322.664844 0.0245269 0.145145 0.168982 0.8658106
ENSG000000000460 87.682625 -0.1471420 0.257007 -0.572521 0.5669691
ENSG000000000938 0.319167 -1.7322890 3.493601 -0.495846 0.6200029
      padj
      <numeric>
ENSG000000000003 0.163035
ENSG000000000005 NA
ENSG000000000419 0.176032
ENSG000000000457 0.961694
ENSG000000000460 0.815849
ENSG000000000938 NA

```

6. Data Visualization

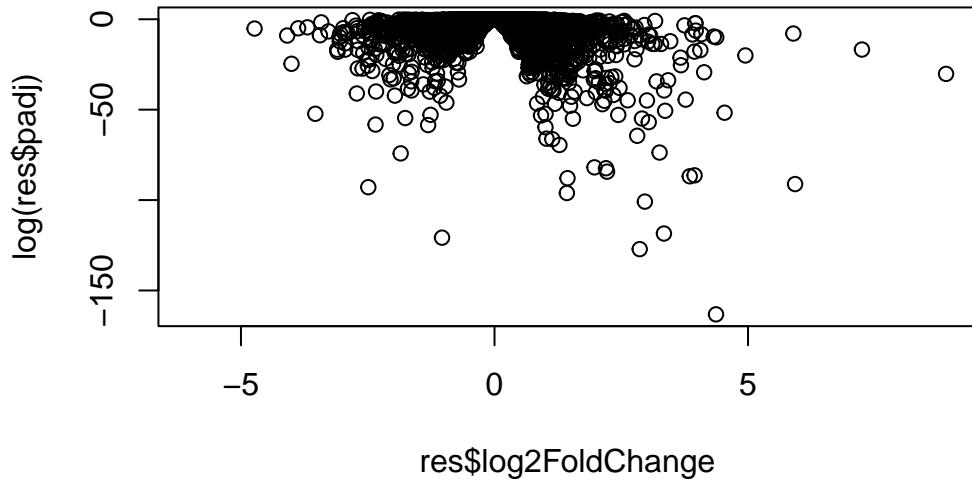
A first plot

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



Well that plot sucked, all the interesting P-values are down below zero. I am going to take the log of the p-value

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

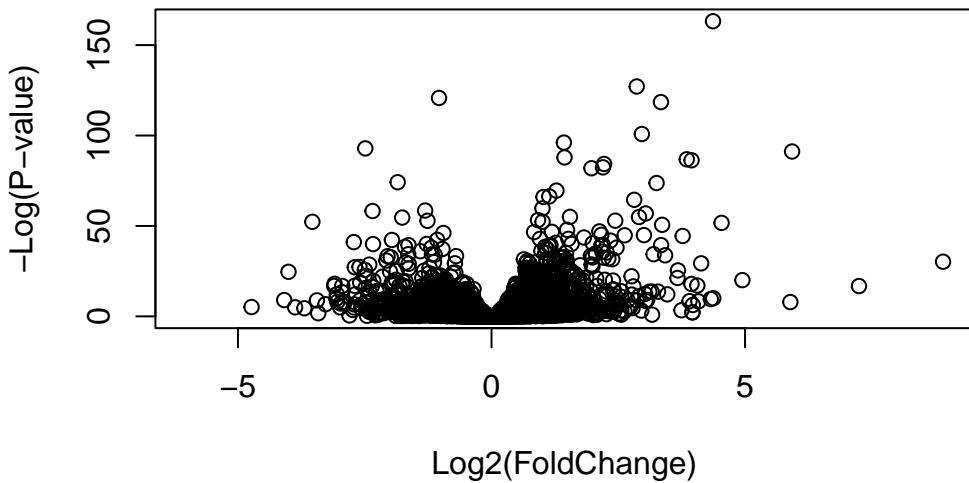


```
log(0.05)
```

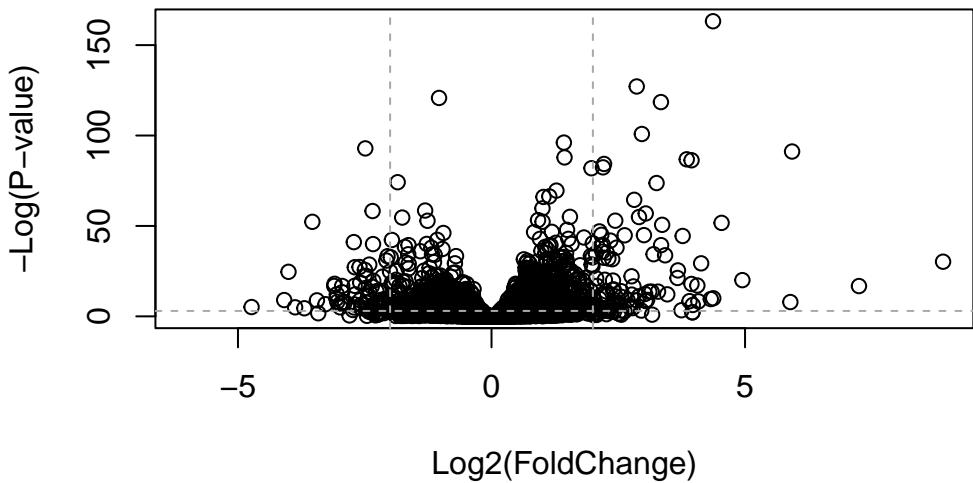
```
[1] -2.995732
```

We can flip the y-axis so the plot does not look “upside down”.

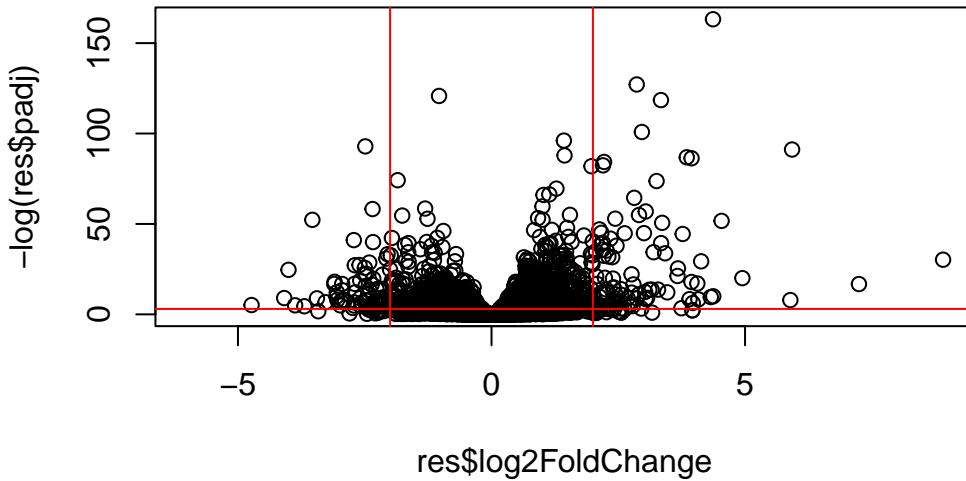
```
plot( res$log2FoldChange, -log(res$padj),
      xlab="Log2(FoldChange)",
      ylab="-Log(P-value)")
```



```
plot( res$log2FoldChange, -log(res$padj),  
      ylab="-Log(P-value)", xlab="Log2(FoldChange)")  
  
# Add some cut-off lines  
abline(v=c(-2,2), col="darkgray", lty=2)  
abline(h=-log(0.05), col="darkgray", lty=2)
```



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

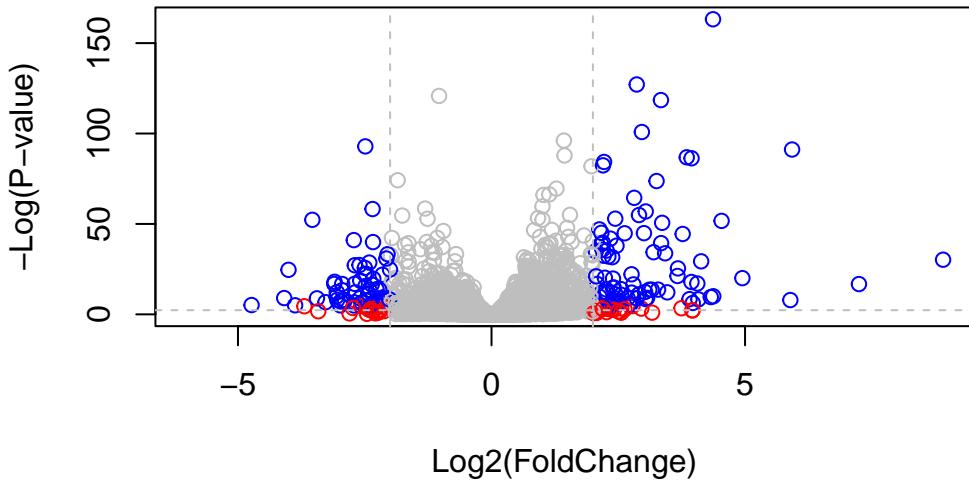


```
# Setup our custom point color vector
mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ] <- "red"

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

# 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="gray", lty=2)
abline(h=-log(0.1), col="gray", lty=2)
```



5. Adding annotation data

Annotation of our gene set results

I will start by loading two Annotation packages from bioconductor:

```
library("AnnotationDbi")
library("org.Hs.eg.db")
```

The `mapIDs()` function “maps” database identifiers between different databases. In other words, it translates the identifiers used by one database to that used by another database.

Let’s see what databases are available for Human data.

```
columns(org.Hs.eg.db)
```

```
[1] "ACNUM"          "ALIAS"          "ENSEMBL"         "ENSEMLPROT"      "ENSEMLTRANS"
[6] "ENTREZID"       "ENZYME"         "EVIDENCE"        "EVIDENCEALL"     "GENENAME"
```

```
[11] "GENETYPE"      "GO"          "GOALL"        "IPI"          "MAP"
[16] "OMIM"          "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"
[21] "PMID"          "PROSITE"      "REFSEQ"       "SYMBOL"      "UCSCKG"
[26] "UNIPROT"
```

My results are in the object `res`.

```
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>
ENSG000000000003 747.194195 -0.3507030 0.168246 -2.084470 0.0371175
ENSG000000000005 0.000000      NA       NA       NA       NA
ENSG000000000419 520.134160  0.2061078 0.101059  2.039475 0.0414026
ENSG000000000457 322.664844  0.0245269 0.145145  0.168982 0.8658106
ENSG000000000460 87.682625 -0.1471420 0.257007 -0.572521 0.5669691
ENSG000000000938 0.319167 -1.7322890 3.493601 -0.495846 0.6200029
  padj
  <numeric>
ENSG000000000003 0.163035
ENSG000000000005  NA
ENSG000000000419 0.176032
ENSG000000000457 0.961694
ENSG000000000460 0.815849
ENSG000000000938  NA
```

```
res$symbol <- mapIds(org.Hs.eg.db,
                      keys=row.names(res), # Our genenames
                      keytype="ENSEMBL",      # The format of our genenames
                      column="SYMBOL",       # The new format we want to add
                      multiVals="first")
```

```
'select()' returned 1:many mapping between keys and columns
```

```
head(res)
```

```

log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 7 columns
  baseMean log2FoldChange    lfcSE      stat     pvalue
  <numeric>    <numeric> <numeric> <numeric> <numeric>
ENSG000000000003 747.194195 -0.3507030 0.168246 -2.084470 0.0371175
ENSG000000000005 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      symbol
  <numeric> <character>
ENSG000000000003 0.163035    TSPAN6
ENSG000000000005      NA      TNMD
ENSG00000000419 0.176032    DPM1
ENSG00000000457 0.961694    SCYL3
ENSG00000000460 0.815849    C1orf112
ENSG00000000938      NA      FGR

```

Add other IDs:

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.

See code run below.

```

res$entrez <- mapIds(org.Hs.eg.db,
  keys=row.names(res), # Our genenames
  keytype="ENSEMBL",      # The format of our genenames
  column="ENTREZID",      # The new format we want to add
  multiVals="first")

'select()' returned 1:many mapping between keys and columns

head(res)

log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 8 columns

```

	baseMean	log2FoldChange	lfcSE	stat	pvalue
	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>
ENSG000000000003	747.194195	-0.3507030	0.168246	-2.084470	0.0371175
ENSG000000000005	0.000000	NA	NA	NA	NA
ENSG000000000419	520.134160	0.2061078	0.101059	2.039475	0.0414026
ENSG000000000457	322.664844	0.0245269	0.145145	0.168982	0.8658106
ENSG000000000460	87.682625	-0.1471420	0.257007	-0.572521	0.5669691
ENSG000000000938	0.319167	-1.7322890	3.493601	-0.495846	0.6200029
	padj	symbol	entrez		
	<numeric>	<character>	<character>		
ENSG000000000003	0.163035	TSPAN6	7105		
ENSG000000000005	NA	TNMD	64102		
ENSG000000000419	0.176032	DPM1	8813		
ENSG000000000457	0.961694	SCYL3	57147		
ENSG000000000460	0.815849	C1orf112	55732		
ENSG000000000938	NA	FGR	2268		

```
res$uniprot <- mapIds(org.Hs.eg.db,
                      keys=row.names(res), # Our genenames
                      keytype="ENSEMBL",      # The format of our genenames
                      column="UNIPROT",       # The new format we want to add
                      multiVals="first")
```

'select()' returned 1:many mapping between keys and columns

```
head(res)
```

log2 fold change (MLE): dex treated vs control
 Wald test p-value: dex treated vs control
 DataFrame with 6 rows and 9 columns

	baseMean	log2FoldChange	lfcSE	stat	pvalue
	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>
ENSG000000000003	747.194195	-0.3507030	0.168246	-2.084470	0.0371175
ENSG000000000005	0.000000	NA	NA	NA	NA
ENSG000000000419	520.134160	0.2061078	0.101059	2.039475	0.0414026
ENSG000000000457	322.664844	0.0245269	0.145145	0.168982	0.8658106
ENSG000000000460	87.682625	-0.1471420	0.257007	-0.572521	0.5669691
ENSG000000000938	0.319167	-1.7322890	3.493601	-0.495846	0.6200029
	padj	symbol	entrez	uniprot	
	<numeric>	<character>	<character>	<character>	

ENSG000000000003	0.163035	TSPAN6	7105	AOA024RCIO
ENSG000000000005	NA	TNMD	64102	Q9H2S6
ENSG000000000419	0.176032	DPM1	8813	060762
ENSG000000000457	0.961694	SCYL3	57147	Q8IZE3
ENSG000000000460	0.815849	C1orf112	55732	AOA024R922
ENSG000000000938	NA	FGR	2268	P09769

```

res$genename <- mapIds(org.Hs.eg.db,
                      keys=row.names(res), # Our genenames
                      keytype="ENSEMBL",      # The format of our genenames
                      column="GENENAME",      # The new format we want to add
                      multiVals="first")

'select()' returned 1:many mapping between keys and columns

head(res)

log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 10 columns
  baseMean log2FoldChange     lfcSE      stat    pvalue
  <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG000000000003 747.194195 -0.3507030  0.168246 -2.084470 0.0371175
ENSG000000000005  0.000000   NA        NA        NA        NA
ENSG000000000419 520.134160  0.2061078  0.101059  2.039475 0.0414026
ENSG000000000457 322.664844  0.0245269  0.145145  0.168982 0.8658106
ENSG000000000460  87.682625 -0.1471420  0.257007 -0.572521 0.5669691
ENSG000000000938  0.319167 -1.7322890  3.493601 -0.495846 0.6200029
  padj      symbol     entrez     uniprot
  <numeric> <character> <character> <character>
ENSG000000000003 0.163035   TSPAN6     7105    AOA024RCIO
ENSG000000000005  NA        TNMD      64102   Q9H2S6
ENSG000000000419 0.176032   DPM1      8813    060762
ENSG000000000457 0.961694   SCYL3     57147   Q8IZE3
ENSG000000000460 0.815849   C1orf112  55732   AOA024R922
ENSG000000000938  NA        FGR       2268    P09769
  genename
  <character>
ENSG000000000003  tetraspanin 6
ENSG000000000005  tenomodulin

```

```

ENSG00000000419 dolichyl-phosphate m..
ENSG00000000457 SCY1 like pseudokina..
ENSG00000000460 chromosome 1 open re..
ENSG00000000938 FGR proto-oncogene, ..

ord <- order( res$padj )
#View(res[ord,])
head(res[ord,])

log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 10 columns
  baseMean log2FoldChange    lfcSE      stat     pvalue
  <numeric>      <numeric> <numeric> <numeric>   <numeric>
ENSG00000152583    954.771      4.36836  0.2371268   18.4220 8.74490e-76
ENSG00000179094    743.253      2.86389  0.1755693   16.3120 8.10784e-60
ENSG00000116584   2277.913     -1.03470  0.0650984  -15.8944 6.92855e-57
ENSG00000189221   2383.754      3.34154  0.2124058   15.7319 9.14433e-56
ENSG00000120129   3440.704      2.96521  0.2036951   14.5571 5.26424e-48
ENSG00000148175   13493.920     1.42717  0.1003890   14.2164 7.25128e-46
  padj      symbol     entrez     uniprot
  <numeric> <character> <character> <character>
ENSG00000152583 1.32441e-71    SPARCL1      8404  AOA024RDE1
ENSG00000179094 6.13966e-56     PER1        5187  O15534
ENSG00000116584 3.49776e-53    ARHGEF2      9181  Q92974
ENSG00000189221 3.46227e-52     MAOA        4128  P21397
ENSG00000120129 1.59454e-44    DUSP1        1843  B4DU40
ENSG00000148175 1.83034e-42     STOM        2040  F8VSL7
  genename
  <character>
ENSG00000152583           SPARC like 1
ENSG00000179094           period circadian reg..
ENSG00000116584           Rho/Rac guanine nucl..
ENSG00000189221           monoamine oxidase A
ENSG00000120129           dual specificity pho..
ENSG00000148175           stomatin

```

Pathway Analysis

Pathway analysis (also known as gene set analysis or over-representation analysis), aims to reduce the complexity of interpreting gene lists via mapping the listed genes to known (i.e. an-

notated) biological pathways, processes and functions.

Some major genesets include KEGG, GO, etc. We will use the *gage* package for our first pathway analysis.

Run in your R console (i.e. not your Rmarkdown doc!)

```
BiocManager::install( c("pathview", "gage", "gageData") )
```

```
library(pathview)
```

```
#####
Pathview is an open source software package distributed under GNU General
Public License version 3 (GPLv3). Details of GPLv3 is available at
http://www.gnu.org/licenses/gpl-3.0.html. Particullary, users are required to
formally cite the original Pathview paper (not just mention it) in publications
or products. For details, do citation("pathview") within R.
```

The pathview downloads and uses KEGG data. Non-academic uses may require a KEGG license agreement (details at <http://www.kegg.jp/kegg/legal.html>).

```
#####
```

```
library(gage)
```

```
library(gageData)
```

```
data(kegg.sets.hs)
```

Examine the first 2 pathways in this kegg set for humans

We can have a look at the first few pathways in the kegg data set.

```
head(kegg.sets.hs, 2)
```

```
$`hsa00232 Caffeine metabolism`  
[1] "10"    "1544"  "1548"  "1549"  "1553"  "7498"  "9"  
  
$`hsa00983 Drug metabolism - other enzymes`  
[1] "10"    "1066"  "10720" "10941" "151531" "1548"  "1549"  "1551"  
[9] "1553"  "1576"  "1577"  "1806"  "1807"  "1890"  "221223" "2990"  
[17] "3251"  "3614"  "3615"  "3704"  "51733"  "54490" "54575"  "54576"  
[25] "54577" "54578" "54579" "54600" "54657"  "54658" "54659"  "54963"  
[33] "574537" "64816" "7083"  "7084"  "7172"  "7363"  "7364"  "7365"  
[41] "7366"  "7367"  "7371"  "7372"  "7378"  "7498"  "79799" "83549"  
[49] "8824"  "8833"  "9"     "978"
```

The main `gaga()` function wants a vector as input that contains our measure of importance - in our case that is fold-change. The vector need to have ENTREZ ids as the names of the vector.

Recall that vectors can have names - this is useful for book-keeping so I know what value corresponds to what value.

```
x <- c(100,80, 10)  
x  
  
[1] 100 80 10  
  
names(x) <- c("Barry", "Lisa", "Chandra")  
x  
  
Barry      Lisa   Chandra  
100       80      10  
  
foldchanges = res$log2FoldChange  
names(foldchanges) = res$entrez  
head(foldchanges)  
  
7105        64102        8813        57147        55732        2268  
-0.35070302          NA  0.20610777  0.02452695 -0.14714205 -1.73228897
```

Now we can run the analysis:

```
# Get the results  
keggres = gage(foldchanges, gsets=kegg.sets.hs)
```

What is in this results object?

```
attributes(keggres)
```

```
$names  
[1] "greater" "less"      "stats"
```

By default gage splits its results into “greater” and “less” objects that you can examine. First we will look at the “less” (i.e.) downregulated pathway results.

```
# Look at the first three down (less) pathways  
head(keggres$less, 3)
```

	p.geomean	stat.mean	p.val
hsa05332 Graft-versus-host disease	0.0004250461	-3.473346	0.0004250461
hsa04940 Type I diabetes mellitus	0.0017820293	-3.002352	0.0017820293
hsa05310 Asthma	0.0020045888	-3.009050	0.0020045888

	q.val	set.size	exp1
hsa05332 Graft-versus-host disease	0.09053483	40	0.0004250461
hsa04940 Type I diabetes mellitus	0.14232581	42	0.0017820293
hsa05310 Asthma	0.14232581	29	0.0020045888

We can now look in more detail at these pathways. The `pathview()` function will take the KEGG pathway ID (printed first above) and our vector of importance and annotate the pathway with our genes.

```
pathview(gene.data=foldchanges, pathway.id="hsa05310")
```

```
'select()' returned 1:1 mapping between keys and columns
```

```
Info: Working in directory /Users/kaitlynpowell/Desktop/BIMM 143/week 6/class12
```

```
Info: Writing image file hsa05310.pathview.png
```

First I will look at hsa05310 Asthma:

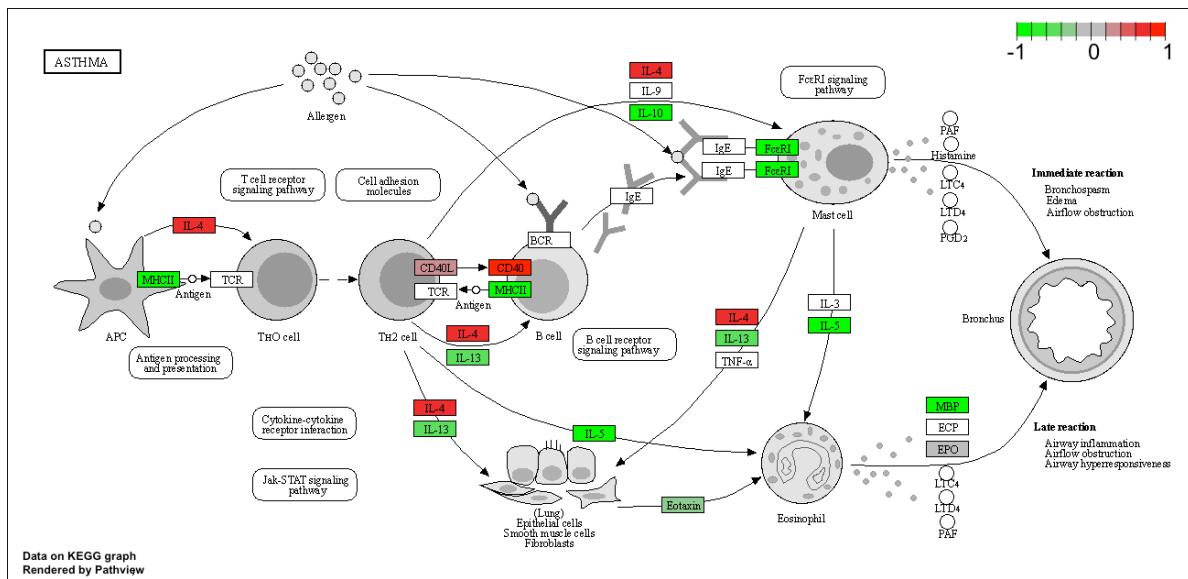


Figure 1: The Asthma pathway with our genes colored

```
# A different PDF based output of the same data
pathview(gene.data=foldchanges, pathway.id="hsa05310", kegg.native=FALSE)
```

```
'select()' returned 1:1 mapping between keys and columns
```

```
Info: Working in directory /Users/kaitlynpowell/Desktop/BIMM 143/week 6/class12
```

```
Info: Writing image file hsa05310.pathview.pdf
```