

Class13: DESeq Lab

Aisha Mohamed

The data for this hands on session comes from a published RNA-seq experiment where airway smooth muscle cells were treated with dexamethasone which is a synthetic glucocorticoid steroid with anti-inflammatory effects.

```
# Complete the missing code  
counts <- read.csv("airway_scaledcounts.csv", row.names=1)  
  
metadata <- read.csv("airway_metadata.csv")  
  
head(counts)
```

	SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516
ENSG000000000003	723	486	904	445	1170
ENSG000000000005	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		
ENSG000000000003	1097	806	604		
ENSG000000000005	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?

```
table(metadata$dex)
```

```
control treated
4           4
```

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

```
[1] 4
```

Toy differential gene expression

Let’s start by calculating the mean counts per gene in the “control” samples. We can then compare this value for each gene to the mean counts in the “treated” samples (i.e. columns).

Step 1: Find which columns in `counts` correspond to “control” samples. Step 2: Calculate the mean value per gene in these columns. Step 3: Store my answer for later in `control.mean`.

```
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

ENSG000000000460	96	81	73	66	118
ENSG000000000938	0	0	1	0	2
	SRR1039517	SRR1039520	SRR1039521		
ENSG000000000003	1097	806	604		
ENSG000000000005	0	0	0		
ENSG000000000419	781	417	509		
ENSG000000000457	447	330	324		
ENSG000000000460	94	102	74		
ENSG000000000938	0	0	0		

```
control inds <- metadata$dex == "control"
```

```
metadata[control inds, ]
```

	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

```
control counts <- counts[, control inds]
```

```
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

```
#apply(control counts, 1, mean)
```

```
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? Is there a function that could help here?

Q4. Follow the same procedure for the treated samples.

Doing the same steps to get `treated.means`.

```

treated inds <- metadata$dex == "treated"
metadata[treated inds, ]

```

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

```

treated counts <- counts[, treated inds]
head(treated counts)

```

	SRR1039509	SRR1039513	SRR1039517	SRR1039521
ENSG000000000003	486	445	1097	604
ENSG000000000005	0	0	0	0
ENSG000000000419	523	371	781	509
ENSG000000000457	258	237	447	324
ENSG000000000460	81	66	94	74
ENSG000000000938	0	0	0	0

```

treated means <- rowMeans(treated counts)
head(treated means)

```

	ENSG000000000003	ENSG000000000005	ENSG000000000419	ENSG000000000457	ENSG000000000460
	658.00	0.00	546.00	316.50	78.75
ENSG000000000938	0.00				

Same code as above but more compact.

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

head(treated.ins)

ENSG000000000003 ENSG000000000005 ENSG000000000419 ENSG000000000457 ENSG000000000460
       658.00          0.00        546.00        316.50        78.75
ENSG000000000938
       0.00
```

To keep us tidy put `control.mean` and `treated.mean` vectors together as two columns of a new data frame.

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

```
head(meancounts)
```

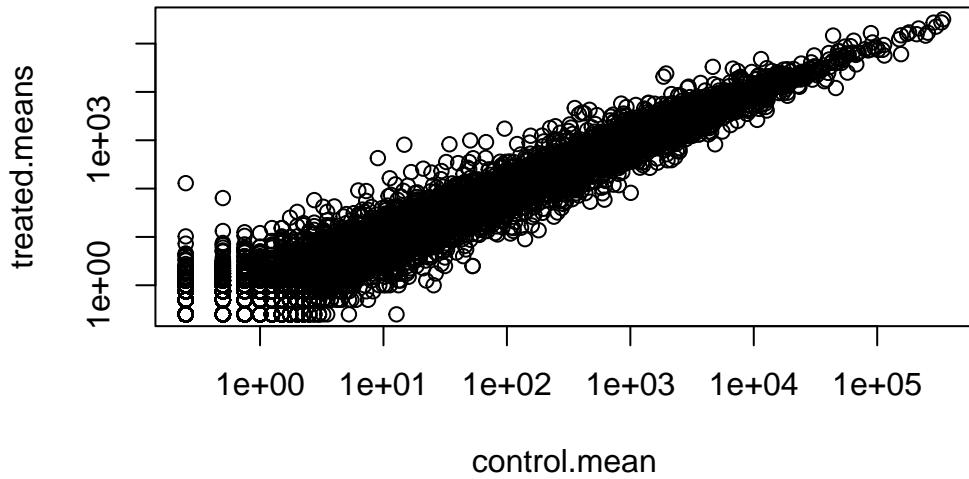
	control.mean	treated.means
ENSG000000000003	900.75	658.00
ENSG000000000005	0.00	0.00
ENSG000000000419	520.50	546.00
ENSG000000000457	339.75	316.50
ENSG000000000460	97.25	78.75
ENSG000000000938	0.75	0.00

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

```
plot(meancounts, log = "xy")
```

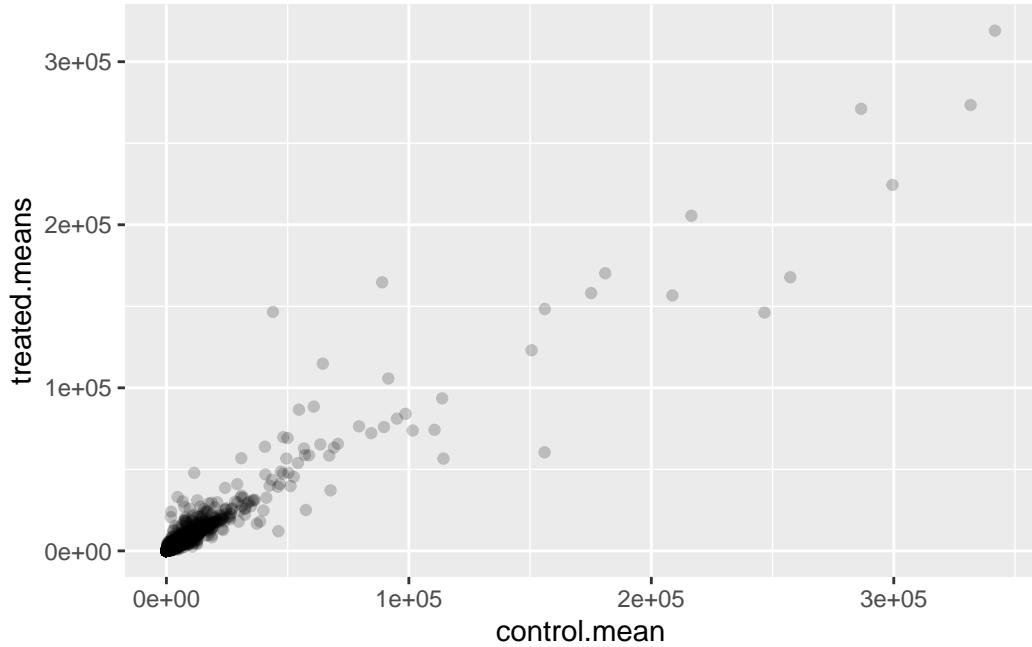
```
Warning in xy.coords(x, y, xlabel, ylabel, log): 15032 x values <= 0 omitted
from logarithmic plot
```

```
Warning in xy.coords(x, y, xlabel, ylabel, log): 15281 y values <= 0 omitted
from logarithmic plot
```



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(control.mean, treated.means) + geom_point(alpha = 0.2)
```

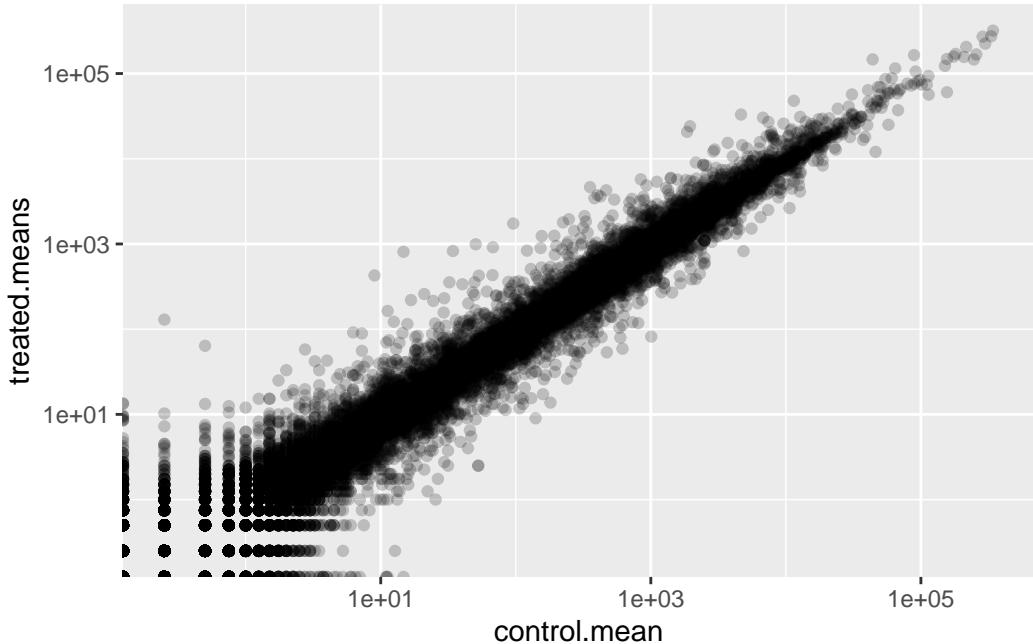


Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

```
ggplot(meancounts) + aes(control.mean, treated.means) +  
  geom_point(alpha = 0.2) + scale_x_log10() + scale_y_log10()
```

Warning: Transformation introduced infinite values in continuous x-axis

Warning: Transformation introduced infinite values in continuous y-axis



Log transformation are super useful when our data is skewed and measured over a wide range like this. We can use different log transformations like base10 or natural logs but we most often prefer log2 units.

```
#Treated/Control
log2(10/10)
```

[1] 0

If there was a doubling

```
#Treated/Control
log2(20/10)
```

[1] 1

Half counts

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

```
[1] -1
```

Quadrupled

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

```
[1] 2
```

Log10 harder to understand.

```
log10(40/10)
```

```
[1] 0.60206
```

Let's add a log2 fold-change column to our little `meancounts` data.frame.

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

head(meancounts)
```

	control.mean	treated.means	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

There are a couple of “weird” results. Namely, the NaN (“not a number”) and -Inf (negative infinity) results.

The NaN is returned when you divide by zero and try to take the log. The -Inf is returned when you try to take the log of zero. It turns out that there are a lot of genes with zero expression. Let's filter our data to remove these genes.

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

#The `!` mark flips the TRUE values to FALSE, and vice versa.
```

```

mycounts <- meancounts[!to.rm.ind,]

dim(mycounts)

[1] 21817      3

head(mycounts)

control.mean treated.means      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?

```

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

# to.rm <- unique(zero.vals[,1])
# mycounts <- meancounts[-to.rm,]
# head(mycounts)

```

The arr.ind argument will utilize the which() function to give back the row/columns which are TRUE values or have Zeros. The unique() function will make sure that rows are repeatedly counted.

A common threshold used for calling something differentially expressed is a log2(FoldChange) of greater than 2 or less than -2. Let's filter the data-set both ways to see how many genes are up or down-regulated.

```

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?

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

```
sum(up.ind)
```

```
[1] 250
```

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

```
[1] 367
```

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

These results can be misleading, further analysis needs to be done to see whether or not these results are significant or not.

Setting up for DESeq

We now need to do statistical significance of these differences.

We will use the DESeq2 package to this analysis properly.

```
library(DESeq2)
```

```
Loading required package: S4Vectors
```

```
Loading required package: stats4
```

```
Loading required package: BiocGenerics
```

```
Attaching package: 'BiocGenerics'
```

```
The following objects are masked from 'package:stats':
```

```
IQR, mad, sd, var, xtabs
```

```
The following objects are masked from 'package:base':
```

```
anyDuplicated, aperm, append, as.data.frame, basename, cbind,
colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
table, tapply, union, unique, unsplit, which.max, which.min
```

```
Attaching package: 'S4Vectors'
```

```
The following object is masked from 'package:utils':
```

```
findMatches
```

```
The following objects are masked from 'package:base':
```

```
expand.grid, I, unname
```

```
Loading required package: IRanges
```

```
Loading required package: GenomicRanges
```

```
Loading required package: GenomeInfoDb
```

```
Loading required package: SummarizedExperiment
```

```
Loading required package: MatrixGenerics
```

```
Loading required package: matrixStats
```

```
Attaching package: 'MatrixGenerics'
```

```
The following objects are masked from 'package:matrixStats':
```

```
colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
colWeightedMeans, colWeightedMedians, colWeightedSds,
colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
rowWeightedSds, rowWeightedVars
```

```
Loading required package: Biobase
```

```
Welcome to Bioconductor
```

```
Vignettes contain introductory material; view with
'browseVignettes()'. To cite Bioconductor, see
'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
Attaching package: 'Biobase'
```

```
The following object is masked from 'package:MatrixGenerics':
```

```
rowMedians
```

```
The following objects are masked from 'package:matrixStats':
```

```
anyMissing, rowMedians
```

```
Set up the object for dds.
```

```
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 we can run our DESeq analysis.

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

estimating size factors

estimating dispersions

gene-wise dispersion estimates

mean-dispersion relationship

final dispersion estimates

fitting model and testing

Get our results back from the `dds` object.

```
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
ENSG00000000419	520.134160	0.2061078	0.101059	2.039475	0.0414026

```

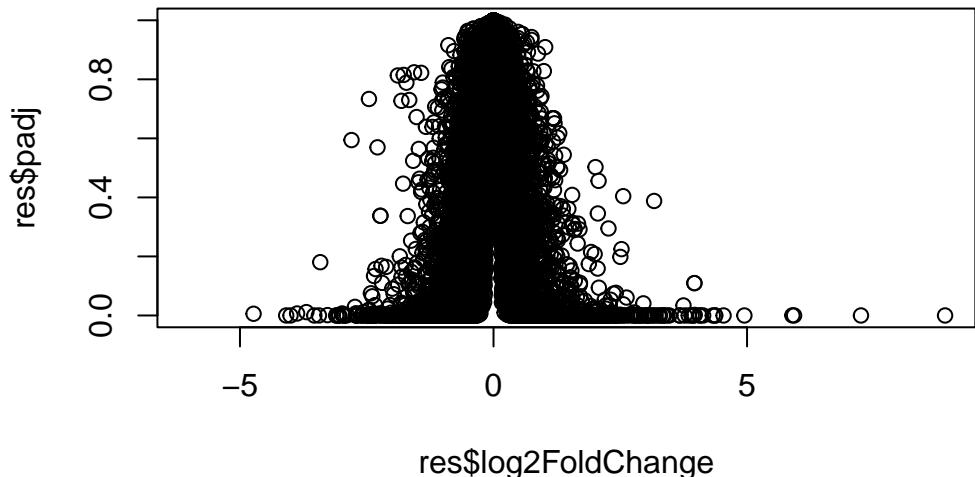
ENSG000000000457 322.664844      0.0245269  0.145145  0.168982  0.8658106
ENSG000000000460  87.682625     -0.1471420  0.257007 -0.572521  0.5669691
ENSG000000000938   0.319167     -1.7322890  3.493601 -0.495846  0.6200029
                    padj
                    <numeric>
ENSG000000000003   0.163035
ENSG000000000005      NA
ENSG000000000419   0.176032
ENSG000000000457   0.961694
ENSG000000000460   0.815849
ENSG000000000938      NA

```

A summary results plot

Volcano plot. This is a common type of summary figure that keeps both our inner biologist and inner stats nerd happy because it shows both P-values and $\log_2(\text{Fold-Changes})$.

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

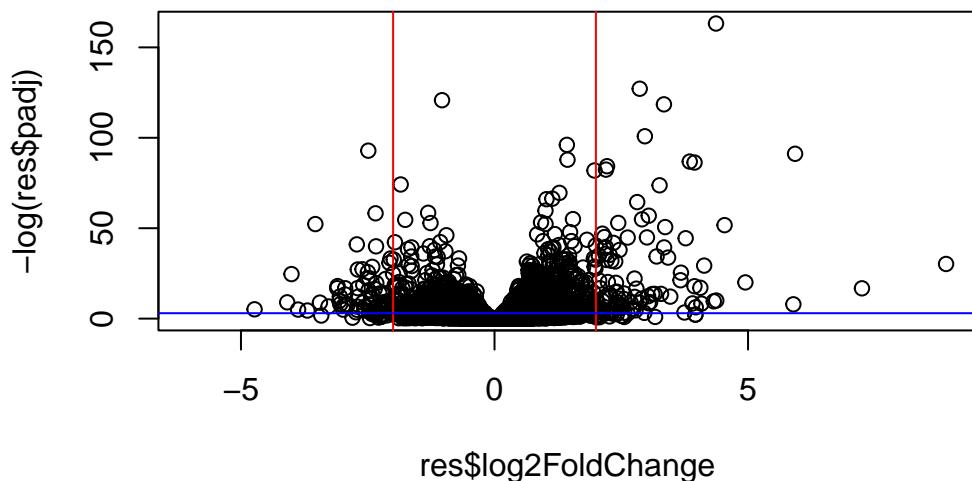


```

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

abline(v = 2, col = "red")
abline(v = -2, col = "red")
abline(h = -log(0.05), col = "blue")

```



Prettier ggplot of the volcano plot.

```

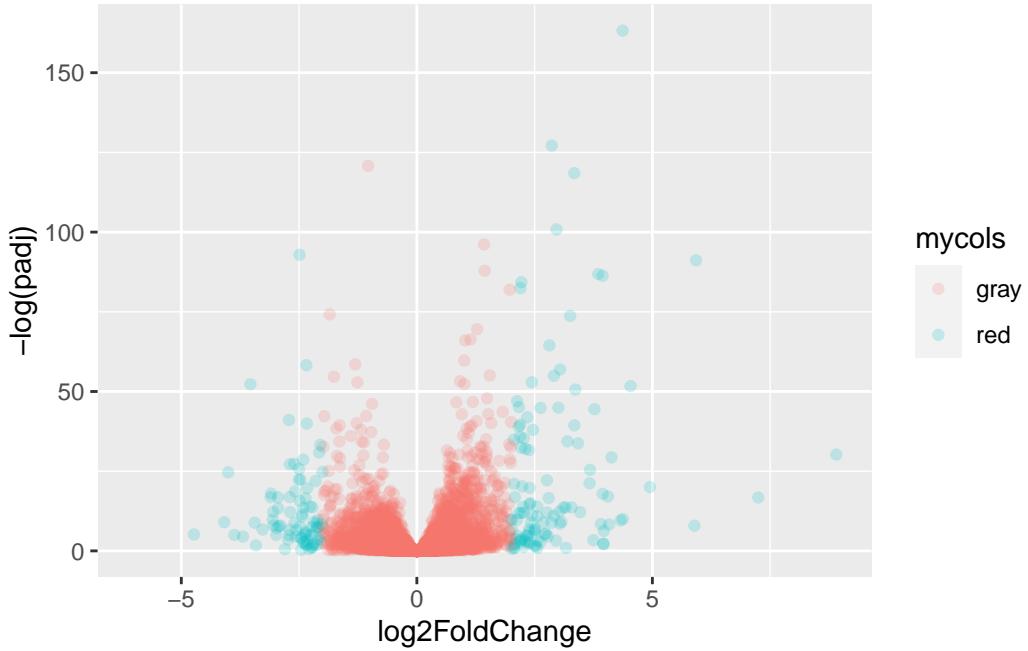
x <- as.data.frame(res)

mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ] <- "red"

ggplot(x) + aes(log2FoldChange, -log(padj), col = mycols) +
  geom_point(alpha = 0.2)

```

Warning: Removed 23549 rows containing missing values (`geom_point()`).



Save our results to date.

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

RNA-Seq DATA pt 2

```
library("AnnotationDbi")
```

Warning: package 'AnnotationDbi' was built under R version 4.3.2

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

To get a list of all available key types that we can use to map between, use the columns() function:

```
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"
```

The main function we will use from the `AnnotationDbi` package is `mapIds()`.

```
head(row.names(res))
```

```
[1] "ENSG000000000003" "ENSG000000000005" "ENSG00000000419" "ENSG00000000457"  
[5] "ENSG00000000460" "ENSG00000000938"
```

These are in ENSEMBL format.

```
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  
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			
	<numeric>	<character>			
ENSG000000000003	0.163035	TSPAN6			
ENSG000000000005	NA	TNMD			
ENSG000000000419	0.176032	DPM1			
ENSG000000000457	0.961694	SCYL3			
ENSG000000000460	0.815849	FIRRM			
ENSG000000000938	NA	FGR			

```
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 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      genename
      <numeric> <character> <character>
ENSG000000000003 0.163035    TSPAN6      tetraspanin 6
ENSG000000000005  NA         TNMD       tenomodulin
ENSG000000000419 0.176032    DPM1       dolichyl-phosphate m..
ENSG000000000457 0.961694    SCYL3      SCY1 like pseudokina..
ENSG000000000460 0.815849    FIRRM      FIGNL1 interacting r..
ENSG000000000938  NA         FGR        FGR proto-oncogene, ..
```

```

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 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      genename entrez
  <numeric> <character> <character> <character>
ENSG000000000003 0.163035    TSPAN6      tetraspanin 6      7105
ENSG000000000005  NA          TNMD       tenomodulin    64102
ENSG000000000419 0.176032    DPM1      dolichyl-phosphate m.. 8813
ENSG000000000457 0.961694    SCYL3      SCY1 like pseudokina.. 57147
ENSG000000000460 0.815849    FIRRM      FIGNL1 interacting r.. 55732
ENSG000000000938  NA          FGR       FGR proto-oncogene, .. 2268

```

Pathway Analysis

We will use the package “gage” along with the “pathview” here to do geneset enrichment (aka pathway analysis) and figure generation respectively.

```
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
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"
```

What we want from `gage()` is our gene in ENTREZ id format with a measure of their importance.

It wants a vector of e.g fold-changes.

```
foldchanges <- res$log2FoldChange  
head(foldchanges)
```

```
[1] -0.35070302           NA  0.20610777  0.02452695 -0.14714205 -1.73228897
```

Add ENTREZ Id's as `names()` to our `foldchange` vector.

```
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 `gage()` pathway analysis.

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

Look at the results.

```
attributes(keggres)
```

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

```
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 view these pathways with our geneset genes highlighted using the `pathview()` function.
For Asthma I will use pathway.id hsa05310 from above.

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

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

Info: Working in directory /Users/aishamohamed/Documents/BIMM 143/Class13

Info: Writing image file hsa05310.pathview.png

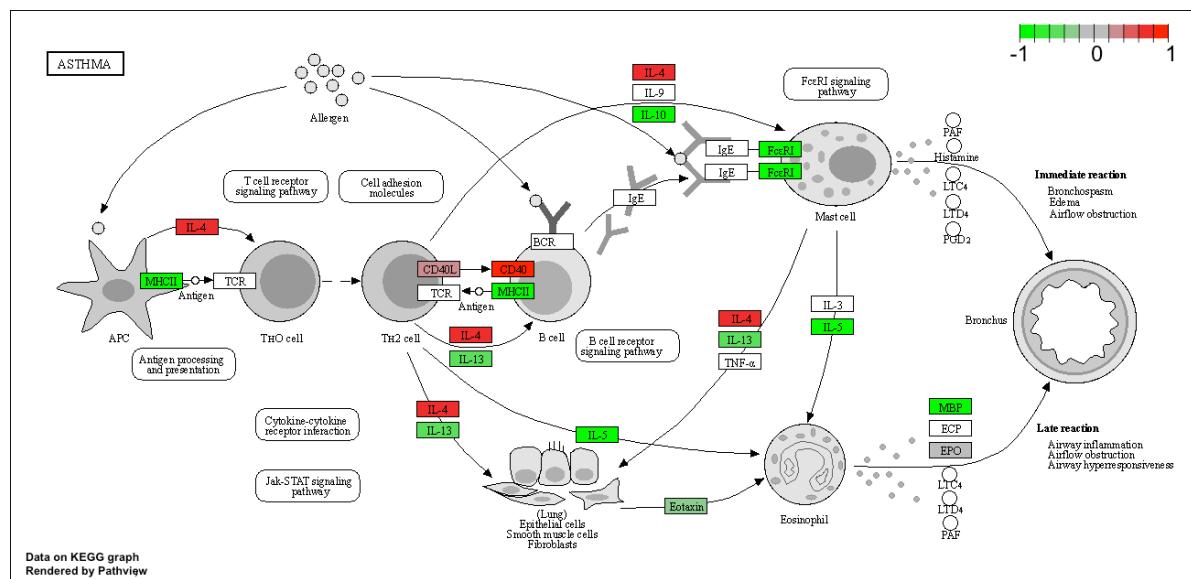


Figure 1: My genes involved in Asthma Pathway