# Class 12

Brianna Smith

## RNAseq Analysis

The data for this hands-on session comes from a published RNA-seq experiment where airway smooth muscle cells were treated with dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al. 2014).

#Import the data

Begin a new R script and use the read.csv() function to read these count data and metadata files.

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

Now, take a look at each.

```
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 |

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

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

4

```
ncol(counts)
```

```
[1] 8
```

```
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
```

Let's make sure that the id column of the metadata match the order of the counts data

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

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

We can use the all function to check `all()` it's inputs are true

```
!all(c(T,T,T,F))
```

```
[1] TRUE
```

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

```
[1] TRUE
```

## Analysis by Hand

```
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
```

library(dplyr) control <- metadata %>% filter(dex==“control”) control.counts <- counts %>% select(control$id) control.mean <- rowSums(control.counts)/4 head(control.mean)

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

Let's first extract our counts for control samples as i want to compare this to our counts fro treated (i.e with drug) samples.

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

I want a summary counts value for each gene in the control experiments. I will start by taking the average.

```
#apply(control.counts, 1, mean)
control.mean <- rowMeans(control.counts)
```

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)

Now do the same for the treated.

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

```
#apply(treated.counts, 1, mean)
treated.mean <- rowMeans(treated.counts)
```

To help us stay organized let's make a new data.frame to store these results together.

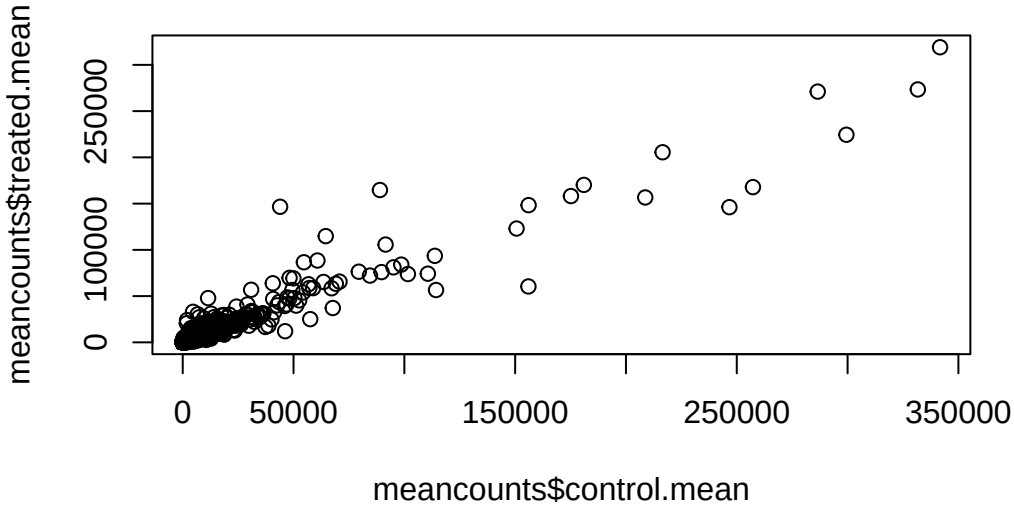
```
meancounts <- data.frame(control.mean, treated.mean)
head(meancounts)
```

```
                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
```

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.

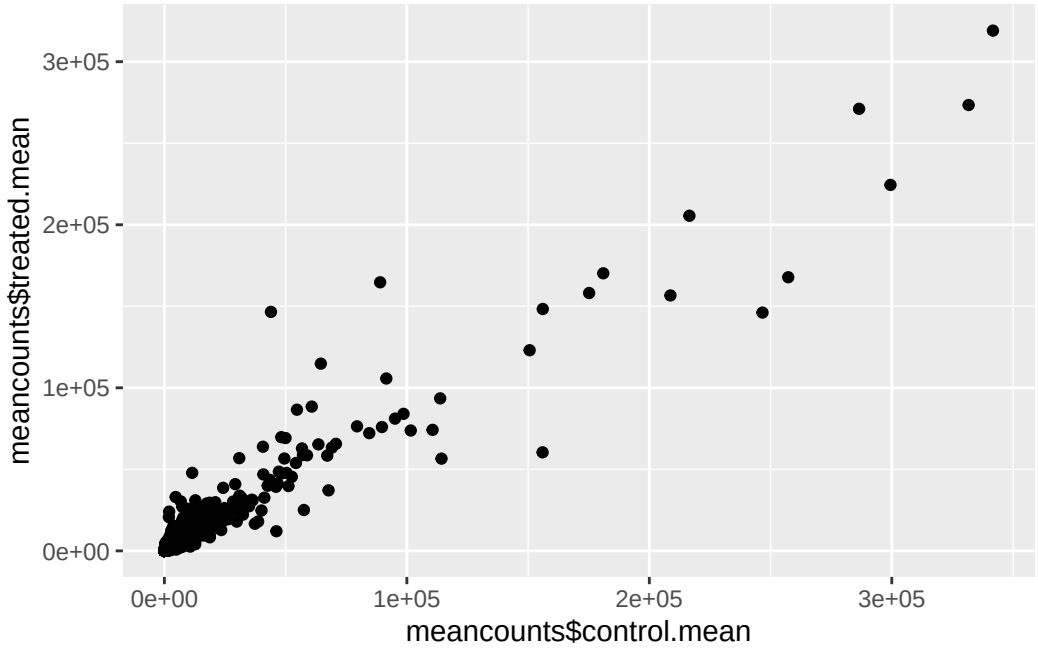
And make a wee plot to see how we are doing.

```
plot(meancounts$control.mean, meancounts$treated.mean)
```




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(meancounts$control.mean, meancounts$treated.mean)) + geom_point()
```




point

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

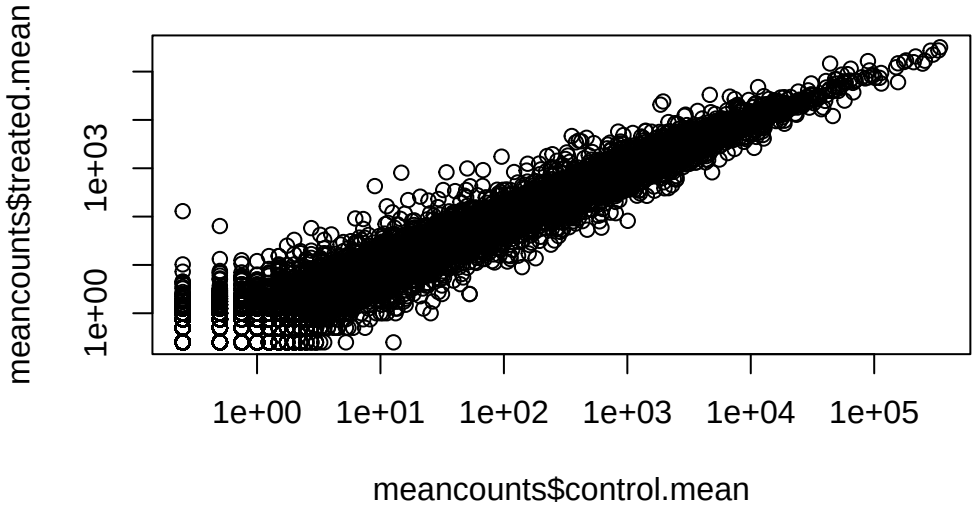
log

This screams for a log transformation

```
plot(meancounts$control.mean, meancounts$treated.mean, 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 the `which()` function to return wherever there is a true value. The `unique()` functions allows for repeated zero values to be counted only once so that there are no duplicates.

The most useful and most straightforward to understand is log2 transform.

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

```
[1] 0
```

Doubling

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

```
[1] 1
```

Half

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

```
[1] -1
```

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

```
[1] 2
```

add a “log2 fold-change”

```
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
```

Hmm… we need to get rid of the genes where we have no count data as taking the log 2 of these 0 counts does not tell us anything.

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

mycounts <- meancounts[to.keep,]
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
```

```
nrow(mycounts)
```

```
[1] 21817
```

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

314

How many genes are up regulated at the log2fc level of +2

```
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?

485

and down regulated...

```
sum(mycounts$log2fc <= -2)
```

```
[1] 485
```

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

We do not trust these results because we don't know if the change is consistent and significant. We are missing stats. Are these big changes significant?

## DESeq2 Analysis

```
library(DESeq2)
```

Like most biocmanager packages DESeq wants it's input in a very specific format.

```
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
```

```
class: DESeqDataSet
dim: 38694 8
metadata(1): version
assays(1): counts
rownames(38694): ENSG00000000003 ENSG00000000005 ... ENSG00000283120
  ENSG00000283123
rowData names(0):
colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
colData names(4): id dex celltype geo_id
```

The main DESeq function 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
```

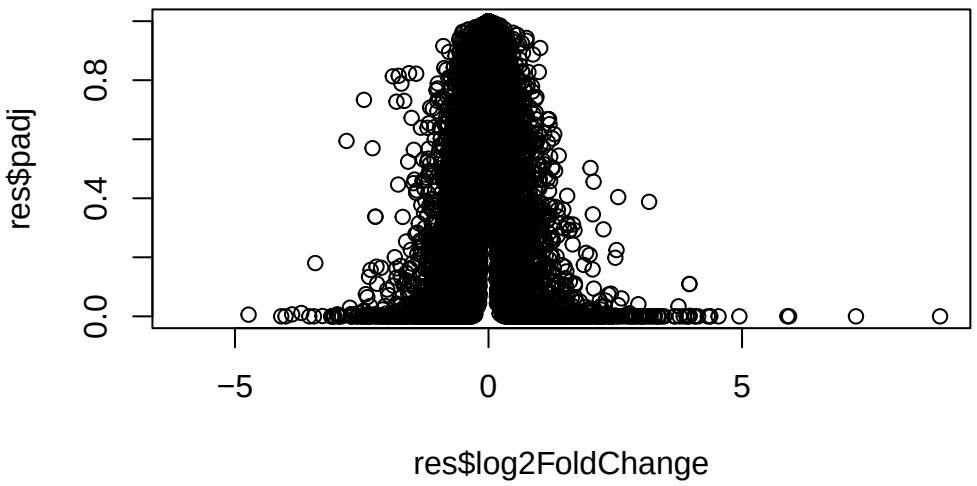
```
res <- results(dds)
head(res)
```

```
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
```

##Volcano Plots

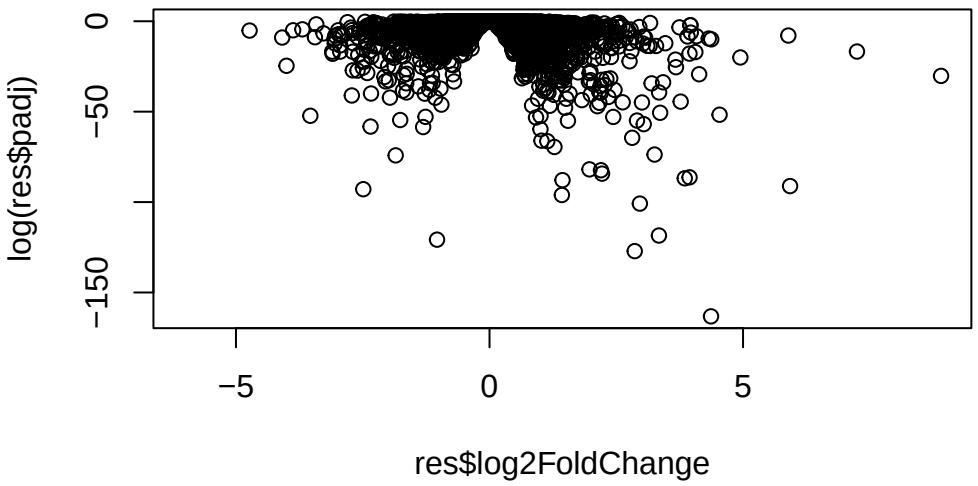
A major summary figure of this type of analysis is called a volcano plot- the idea here is to keep our inner biologist and inner stats person happy with one cool plot!

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



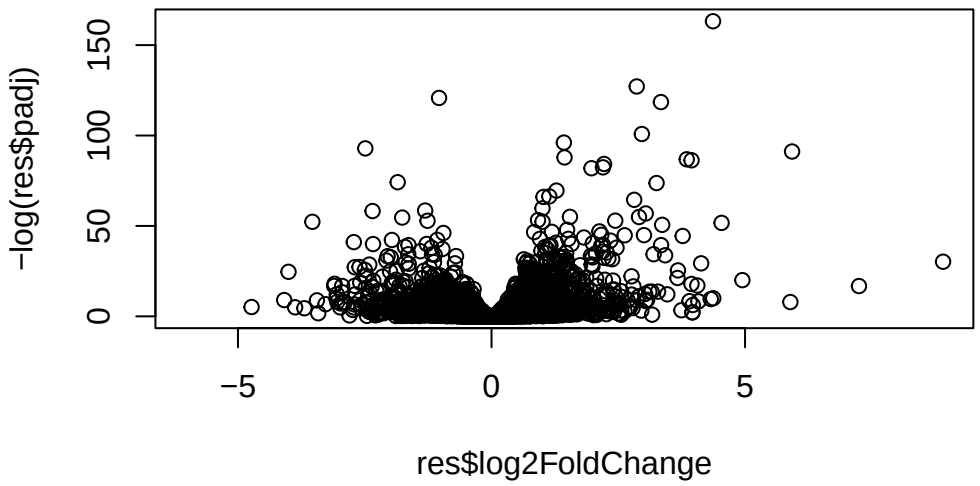

Improve this plot by taking the log of the p-value axis

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




I want to flip the y-axis so the values I care about (i.e the low p-value or high log(p-values)) are at the top of the axis

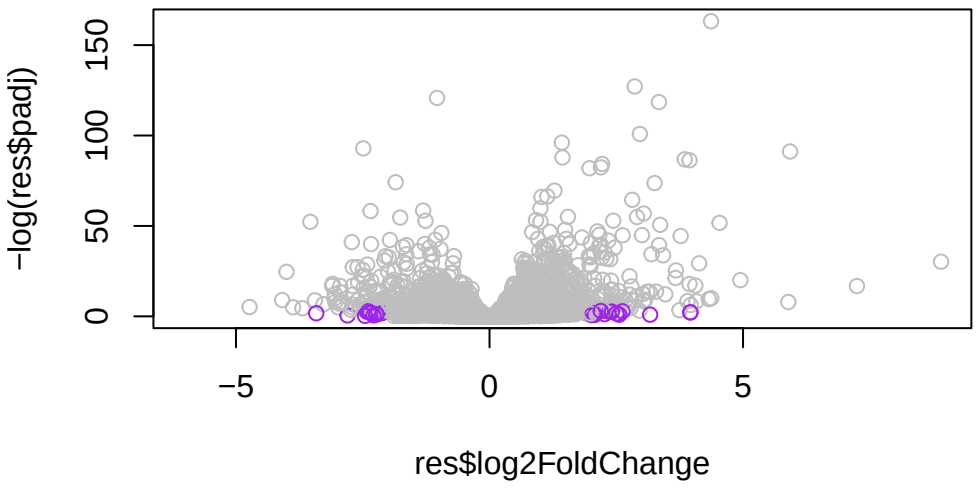
```
plot(res$log2FoldChange, -log(res$padj))
```



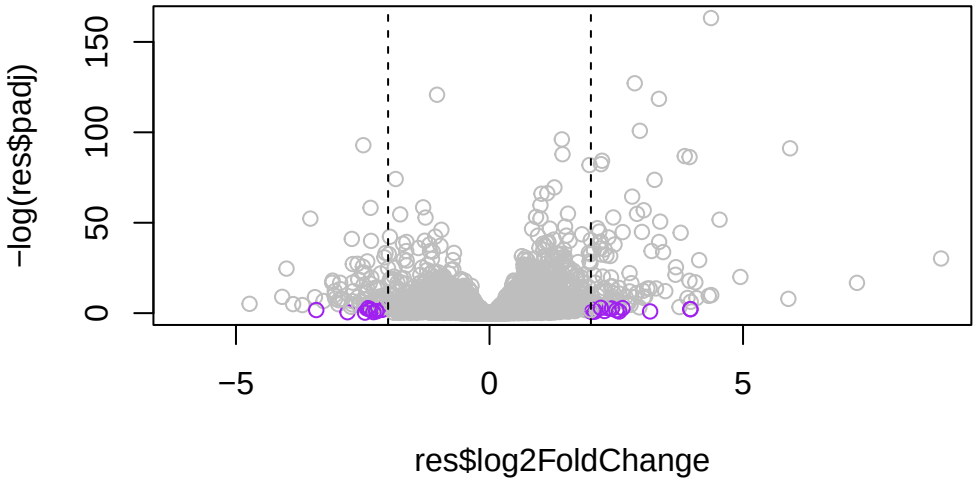

Let's finish up for today by adding some color to better highlight the subset of genes that we will focus on next today - i.e. those with big log2fc values (at +2/-2 threshold) and significant p-values (less than 0.5 for example).

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

```
plot(res$log2FoldChange, -log(res$padj), col=mycols)
```




```
plot(res$log2FoldChange, -log(res$padj), col=mycols)
abline(v=c(-2,2), lty=2)
```



## Gene Annotation

We will use one of Bioconductor's main annotation package to help with mapping between various ID schemes. Here we load the `AnnotationDbi` package and the annotation ata package for humans `org.Hs.eg.db`.

```
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
```

```
#rownames(res)
```

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

Look at what types of IDs I can translate from the `org.Hs.eg.db` packages with the `columns()` function.

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

```
 [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
 [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"
```

```
res$symbol <- mapIds(x=org.Hs.eg.db, column = "SYMBOL", keys = rownames(res), keytype = "E
```

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

and do the same for ENTREZID and GENENAME

```
res$entrez <- mapIds(x= org.Hs.eg.db, column = "ENTREZID", keys = row.names(res), keytype
```

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

```
res$uniprot <- mapIds(x= org.Hs.eg.db, column = "UNIPROT", keys = row.names(res), keytype
```

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

```
res$genename <- mapIds(x= org.Hs.eg.db, column = "GENENAME", keys = row.names(res), keytyp
```

```
'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>
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      symbol      entrez     uniprot
                <numeric> <character> <character> <character>
ENSG00000000003  0.163035      TSPAN6        7105  A0A024RCI0
ENSG00000000005        NA        TNMD       64102      Q9H2S6
ENSG00000000419  0.176032        DPM1        8813      O60762
ENSG00000000457  0.961694       SCYL3       57147      Q8IZE3
ENSG00000000460  0.815849    C1orf112       55732  A0A024R922
ENSG00000000938        NA         FGR        2268      P09769
                           genename
                        <character>
ENSG00000000003          tetraspanin 6
ENSG00000000005            tenomodulin
ENSG00000000419 dolichyl-phosphate m..
ENSG00000000457 SCY1 like pseudokina..
ENSG00000000460 chromosome 1 open re..
ENSG00000000938 FGR proto-oncogene, ..
```

## Pathway Analysis

We will finish this lab with a quick pathway analysis. Here we play with just one, the **GAGE package** (which stands for Generally Applicable Gene set Enrichment), to do **KEGG pathway enrichment analysis**

```
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"
```

The main `gage()` function requires a named vector of fold changes, where the names of the values are the Entrez gene IDs.

```
c(barry=4, clair=3, chandra=2)
```

```
barry   clair chandra
    4       3       2
```

```
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, let's run the gage pathway analysis.

```
keggres = gage(foldchanges, gsets=kegg.sets.hs)
```

Now, let's look at the object returned from `gage()` i.e. our results here:

```
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
```

Let's pulls up the highlighted pathways and show our differently expressed genes on the pathway. i will use the "hsa" KEGG id to get the pathway from KEGG and my `folgchange` vector to show my genes.

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

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

```
Info: Working in directory /Users/briannasmith/Desktop/BIMM 143/Class 12
```

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

Put this into my document.

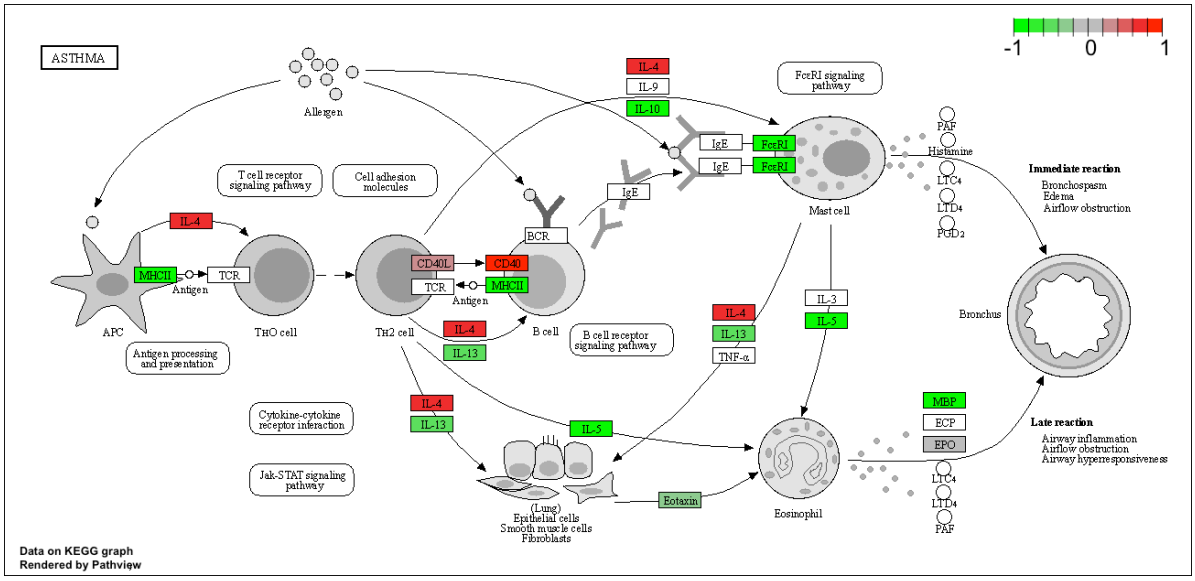



Figure 1: The Asthma pathway with my highlighted differentially expressed genes in color