Class Lab 13: Transcriptomics and the analysis of RNA-Seq data

Nathaniel Nono (PID:A16782656)

Background

Today we will analyze some RNAse data from Himes et al. on the effects of dexmethasone

- Data = published RNA-seq experiment where airway smooth muscle cells (ASM) were treated with dexamethasone
- Dexamethasone (dex) - a synthetic glucocorticoid steroid with anti-inflammatory effects

Bioconductor Setup

Packages to install in console

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

```
BiocManager::install()
```

```
BiocManager::install("DESeq2")
```

Using the package

```
library(BiocManager)
```

```
library(DESeq2)
```

Import countData and colData

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

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

There are 38694 genes in the counts dataset

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

Approach 1

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

```
control treated
      4       4
```

Approach 2

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

```
[1] 4
```

There are 4 controls in this dataset

Toy Differential gene expression

Mean per gene count

Calculate the mean per gene count values for all control samples (i.e. columns in `counts`). Do the same for the 'treated' and then compare them

1. Find all "control" values/columns in `counts` by finding the control values in the meta data

```
control.inds <- metadata$dex == 'control'

# Looks through all counts and access all the TRUE columns
control.counts <- counts[,control.inds]
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
```

2. Find the mean per gene across all control columns

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

```
head(control.mean)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         900.75            0.00          520.50          339.75            97.25
ENSG00000000938
           0.75
```

3. Do the same steps to find the `treated.mean` values

```
treated.inds <- metadata$dex == 'treated'

treated.counts <- counts[,treated.inds]
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 <- apply(treated.counts, 1, mean)
head(treated.mean)
```

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

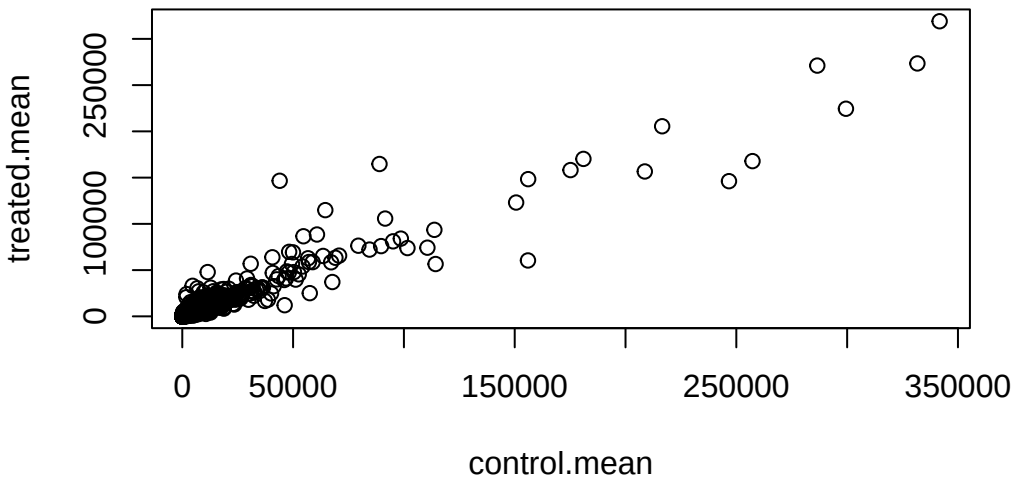
Q3. How would you make the above code in either approach more robust? Is there a function that could help here?

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

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.

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

```
plot(meancounts)
```

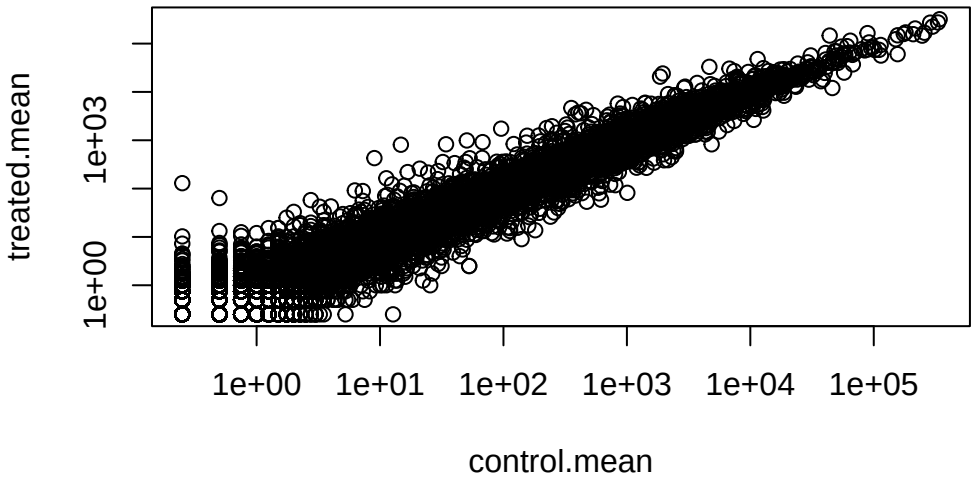


Above data is heavily skewed. Transform the data by taking the log of both x and y

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

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

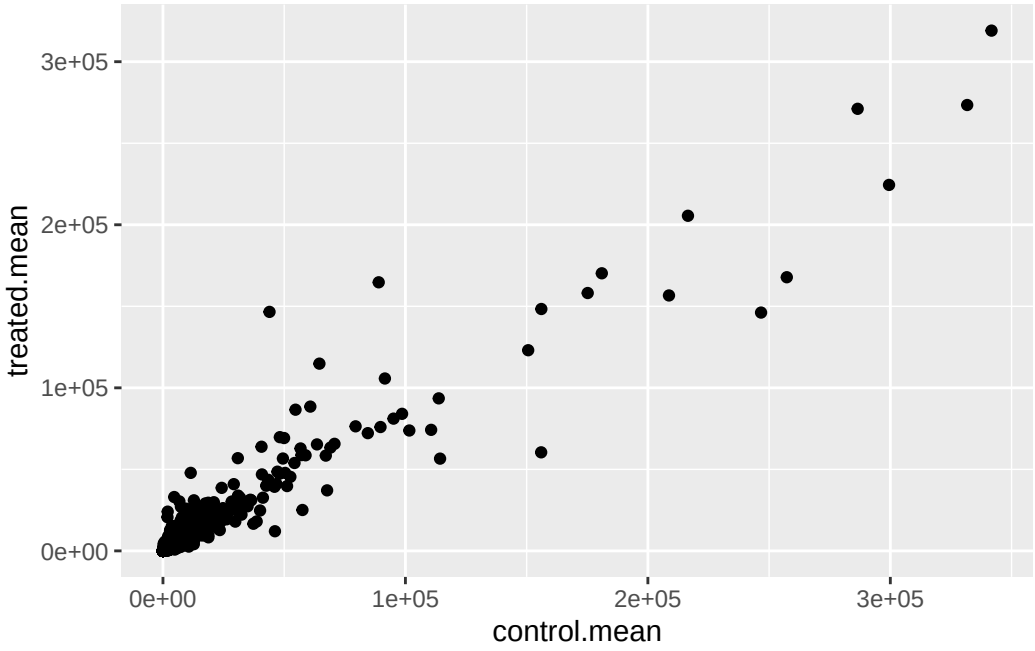
```
Warning in xy.coords(x, y, xlabel, ylabel, log): 15281 y values <= 0 omitted
from logarithmic plot
```



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)
```

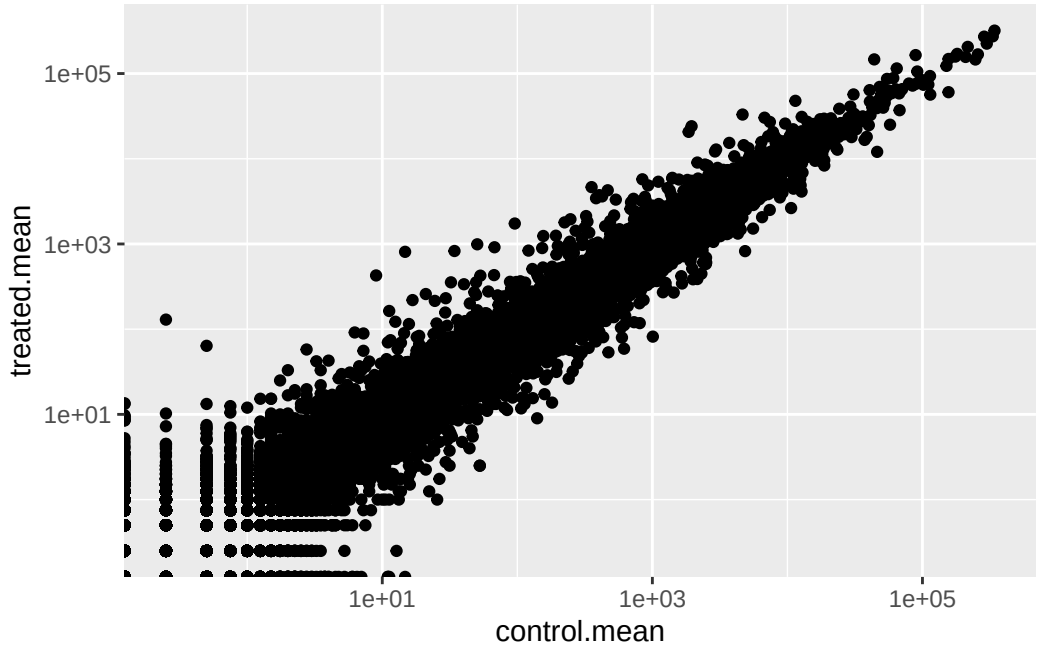
```
ggplot(meancounts) +
  aes(control.mean,
      treated.mean) +
  geom_point()
```




```
ggplot(meancounts) +
  aes(control.mean,
      treated.mean) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10()
```

```
Warning in scale_x_log10(): log-10 transformation introduced infinite values.
```

```
Warning in scale_y_log10(): log-10 transformation introduced infinite values.
```



These log2 values make the interpretation of "fold-change" a little easier and a rule of thumb in the filled is a log2 fold-change of +2 or -2 is where we start to pay attention

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

```
[1] 2
```

Let's calculate the log2(fold-change and add it it to our `meancounts` data.frame)

```
# Add a new column
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
```

Doing this might result in `NaN` or `-inf` values

1. Access the meancounts
2. Are any of the values 0?
3. Sum the rows

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

How many genes do I have left after this zero count filtering?

```
nrow(mycounts)
```

```
[1] 21817
```

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

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?

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

How many genes are 'up' regulated upon drug treatment using a threshold of +2 log2-fold-change?

1. Extract the log2fc values
2. Find those that are above +2
3. Count them

```
# Give out a logical then sum
sum(mycounts$log2fc > 2)
```

```
[1] 250
```

There are 250 genes that are up-regulated upon drug treatment

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

How many genes are 'down' regulated upon drug treatment using a threshold of -2 log2-fold-change?

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

```
[1] 21332
```

There are 21332 genes that are down-regulated upon drug treatment

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

We still need statistics to ensure that there is significance in the data so we can't completely trust these results just yet

Setting up for DESeq

Performing the analysis the right way with stats and use the `DESeq2` package

```
library(DESeq2)
```

The first function that we will use will setup the data in the way (format) DESeq wants it.

```
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
```

The function in the package is called `DESeq()` and we can run it on our `dds` object

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

```
estimating size factors

estimating dispersions

gene-wise dispersion estimates

mean-dispersion relationship

final dispersion estimates

fitting model and testing
```

DESeq analysis

I will get the results from dds with the `results()` function

```
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
```

Summary of the data

```
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
```

Data visualization using a volaqno

P-values

```
log(0.005)
```

```
[1] -5.298317
```

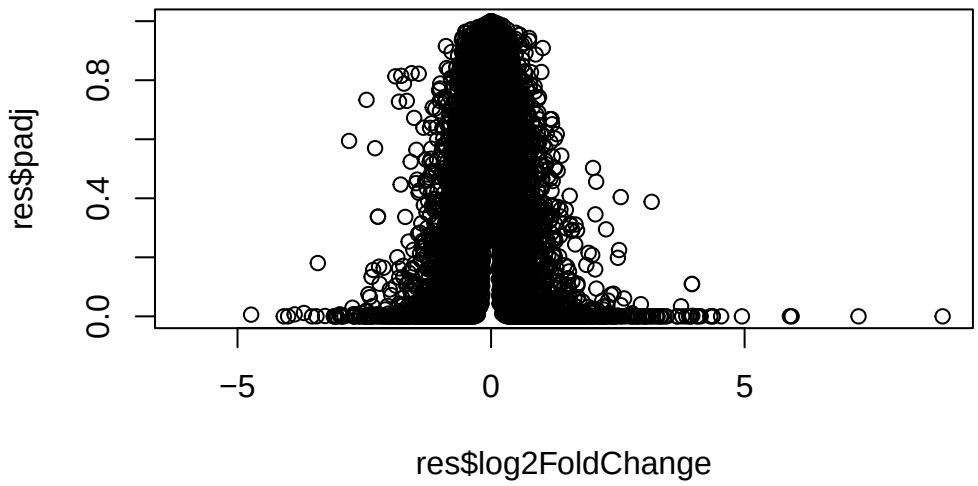
```
log(0.0000001)
```

```
[1] -16.1181
```

Volcano plot

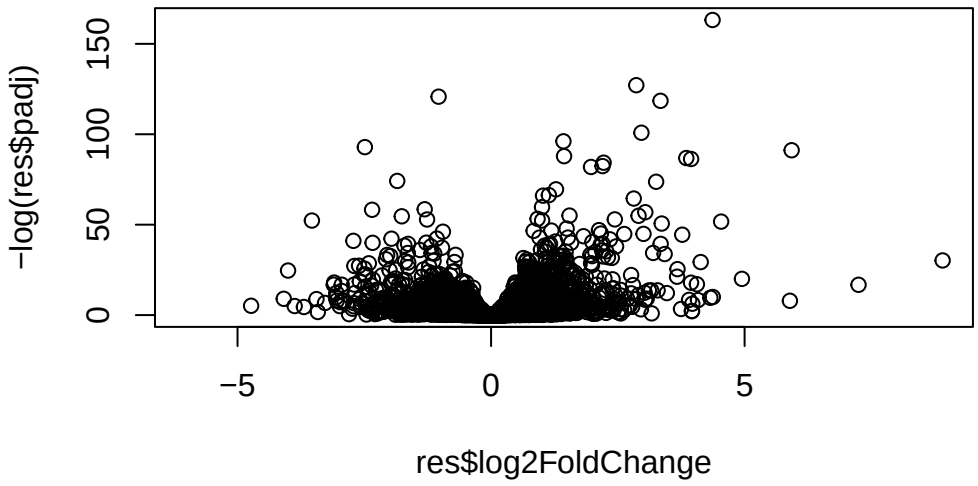
This plot makes a commone overall results figure from this analysis by plotting fold-change vs P-value

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



We care about really small p-values (< 0.05). Let's try transforming the data

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



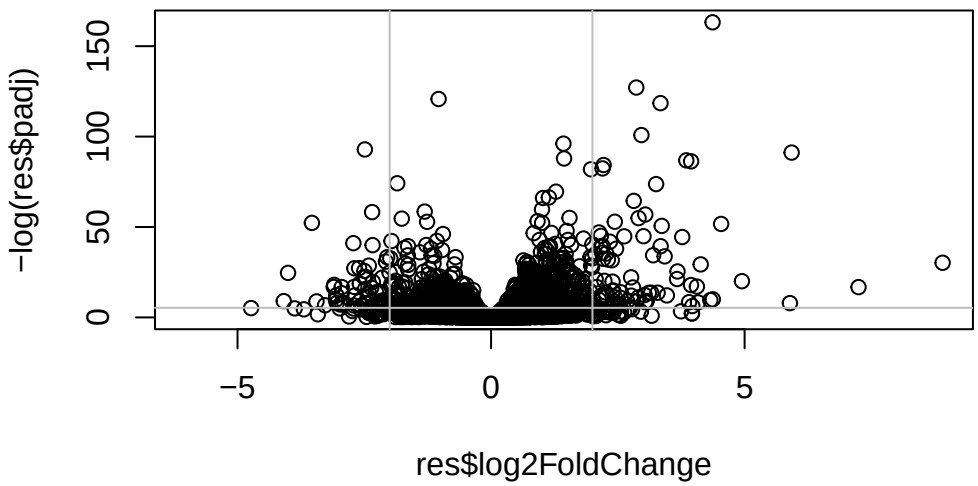
Polishing the plot up

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

# Any point to the right of the line is up regulated
abline(v=2, col = 'gray')

# Any pointto the left of the line is down regulated
abline(v=-2, col = 'gray')

# Showing p-value
abline(h=-log(0.005), col = 'gray')
```

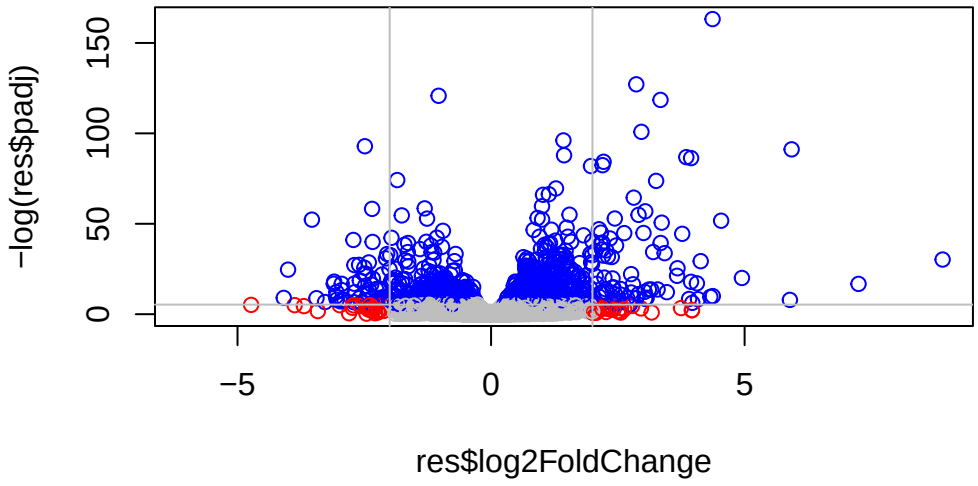


Adding color to the points

```
# Place holder color vector
mycols <- rep('gray', nrow(res))
mycols[res$log2FoldChange > 2] <- 'red'
mycols[res$log2FoldChange < -2] <- 'red'
mycols[res$padj < 0.005] <- 'blue'

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

```
abline(v=-2, col = 'gray')
abline(h=-log(0.005), col = 'gray')
```

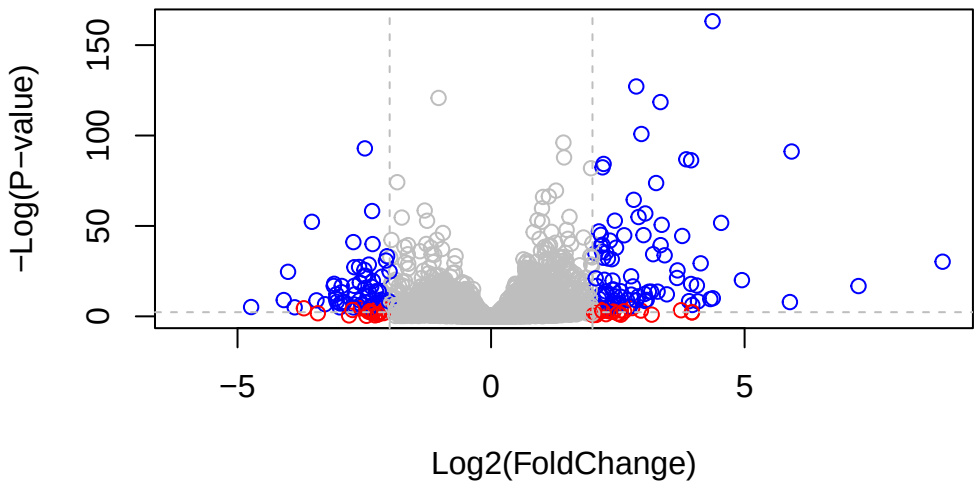



```
# Setup our custom point color vector
mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ]  <- "red"

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)
```



Need to continue on the next day: I want to save my results to date out to disc

```
write.csv(res, file = 'myresults.csv')
```

We will pick this up next day and add **annotation** (i.e. what are these genes of interest) and do **pathway analysis** (what biology) are they known to be involved with.

```
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
```

Adding annotation data

I need to translate our gene identifier "ENSGOOOO..." into gene names that the rest of the world can understand

To this "annotation" I will use the **AnnotationDbi** bioconductor package. Insstall it in the console: `BiocManager::install()`

`BiocManager::install("AnnotationDbi")` `BiocManager::install("org.Hs.eg.db")`

Make sure to do it in the console and type `n`if called to update

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

```
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"
```

I will use the `mapIds()` function to "map" my indetifiers to those from different databses. I will go between "ENSEMBL" and "SYMBOL" (and then after gene name)

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

```
'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>
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
                <numeric> <character>
ENSG00000000003  0.163035      TSPAN6
ENSG00000000005        NA        TNMD
ENSG00000000419  0.176032        DPM1
ENSG00000000457  0.961694       SCYL3
ENSG00000000460  0.815849       FIRRM
ENSG00000000938        NA         FGR
```

ADD “GENENAME”

```
res$genename <- mapIds(org.Hs.eg.db,
       keys = rownames(res),
       keytype = "ENSEMBL",
       column = "GENENAME")
```

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

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

```
'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>
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               genename      entrez
                <numeric> <character>            <character> <character>
ENSG00000000003  0.163035      TSPAN6          tetraspanin 6        7105
ENSG00000000005        NA        TNMD            tenomodulin       64102
ENSG00000000419  0.176032        DPM1 dolichyl-phosphate m..        8813
ENSG00000000457  0.961694       SCYL3 SCY1 like pseudokina..       57147
ENSG00000000460  0.815849       FIRRM FIGNL1 interacting r..       55732
ENSG00000000938        NA         FGR FGR proto-oncogene, ..        2268
```

Save our annotated results object

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

```
res_an <- read.csv("results_annotated")
```

```
colnames(res_an)
```

```
 [1] "X"              "baseMean"       "log2FoldChange" "lfcSE"
 [5] "stat"           "pvalue"         "padj"           "symbol"
 [9] "genename"       "entrez"
```

```
head(res_an)
```

```
                X    baseMean log2FoldChange     lfcSE       stat     pvalue
1 ENSG00000000003 747.1941954    -0.35070302 0.1682457 -2.0844697 0.03711747
2 ENSG00000000005   0.0000000             NA        NA         NA         NA
```

```
3 ENSG00000000419 520.1341601     0.20610777 0.1010592  2.0394752 0.04140263
4 ENSG00000000457 322.6648439     0.02452695 0.1451451  0.1689823 0.86581056
5 ENSG00000000460  87.6826252    -0.14714205 0.2570073 -0.5725210 0.56696907
6 ENSG00000000938   0.3191666    -1.73228897 3.4936010 -0.4958463 0.62000288
       padj symbol                                                    genename
1 0.1630348 TSPAN6                                                tetraspanin 6
2        NA   TNMD                                                  tenomodulin
3 0.1760317   DPM1 dolichyl-phosphate mannosyltransferase subunit 1, catalytic
4 0.9616942  SCYL3                                     SCY1 like pseudokinase 3
5 0.8158486  FIRRM   FIGNL1 interacting regulator of recombination and mitosis
6        NA    FGR              FGR proto-oncogene, Src family tyrosine kinase
  entrez
1   7105
2  64102
3   8813
4  57147
5  55732
6   2268
```

Pathway analysis

Now that we have our results with added annotation we can do some pathway mapping

Let's use the **gage** package to look for KEGG pathways in our results (genes of interest). I will also use the **pathview** package to draw little pathway figures

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

```
library(gage)
```

```
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(gageData)

data(kegg.sets.hs)

# Examine the first 2 pathways in this kegg set for humans
head(kegg.sets.hs, 1)
```

```
$`hsa00232 Caffeine metabolism`
[1] "10"   "1544" "1548" "1549" "1553" "7498" "9"
```

What **gage** wants as inpur is not my big table/data.frame of results. It just wants a "vector of importance". For RNASeq data like we have this is our log2FC values...

```
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

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

What is in this `keggres`

```
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 use the pathview package to look at one of these highlighted KEGG pathways with our genes highlighted. "hsa05310 Asthma"

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

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

```
Info: Working in directory /Users/nathaniel/Desktop/Academia/2024-2025/Fall/Bioinformatics/cl
```

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

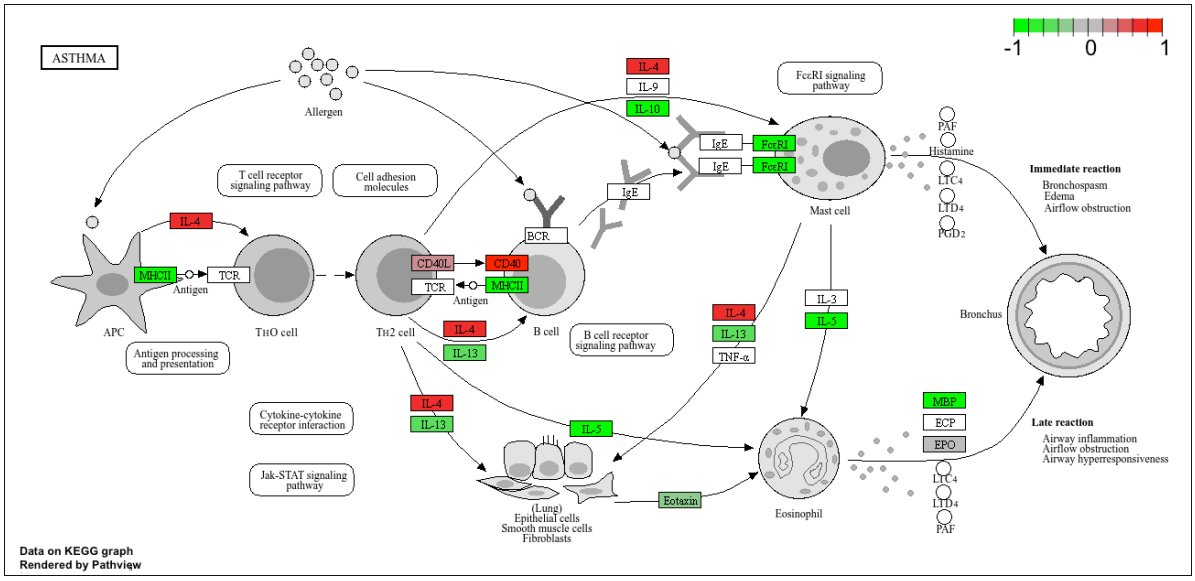


Figure 1: Asthma pathway with my DEGs

Q12. Can you do the same procedure as above to plot the pathview figures for the top 2 down-reguled pathways?