Class 13: RNASeq with DESeq

Aadhya Tripathi (PID: A17878439)

Table of contents

Background

Today, we will perform and RNASeq analysis on the effects of dexamethasone (hereafter "dex"), a common steroid, on airway smooth muscle (ASM) cell lines.

Data Import

We need 2 things for this analysis:

- **countData**: a table with genes as rows and samples/experiments as columns.
- **colData**: metadata about the columns (i.e. samples) in the main countData object.

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

Quick look at metadata:

```
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
```

Quick look at counts:

```
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
```

Check on metadata counts correspondence

We need to check that the metadata matches the samples in our count data.

```
ncol(counts) == nrow(metadata)
```

```
[1] TRUE
```

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

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

```
# can use `all(c([comparisons]))` to return one boolean
```

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

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

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

```
[1] 4
```

Analysis Plan...

We have 4 replicates per condition ("control" and "treated"). We want to compare the control vs the treated to see which genes expression levels change when the drug is present.

For each row (each gene), we will see if the average gene expression value for the control samples is different from the average gene expression value for the treated samples.

1. Find which columns in `counts` corresponse to "control" samples, using the `metadata`
2. Extract/select/etc. the control columns.
3. Calculate the average expression value for each gene (each row).
4. Repeat 1-3 for the "treated" samples.

```
# the indices (i.e. positions) that are "control"
control.inds <- metadata$dex == "control"

# Extract these "control" columns from counts
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
```

```
# Calculate the mean for each row/gene
control.mean <- rowMeans(control.counts)
head(control.mean)
```

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

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

rowSums()

Q4. Do the same for "treated" samples (find the mean count value per gene).

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

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

Let's put these two mean values into a new data.frame `meancounts` for easy book-keeping and plotting.

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

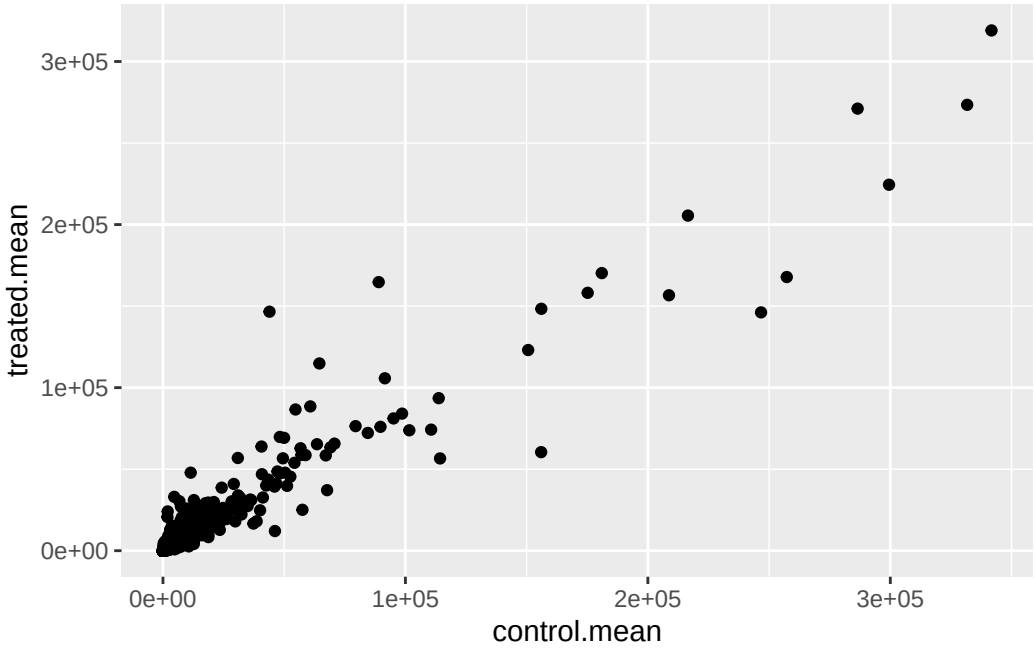
```
                control.mean treated.mean
ENSG00000000003       900.75       658.00
ENSG00000000005         0.00         0.00
```

```
ENSG00000000419       520.50       546.00
ENSG00000000457       339.75       316.50
ENSG00000000460        97.25        78.75
ENSG00000000938         0.75         0.00
```

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

```
library(ggplot2)

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



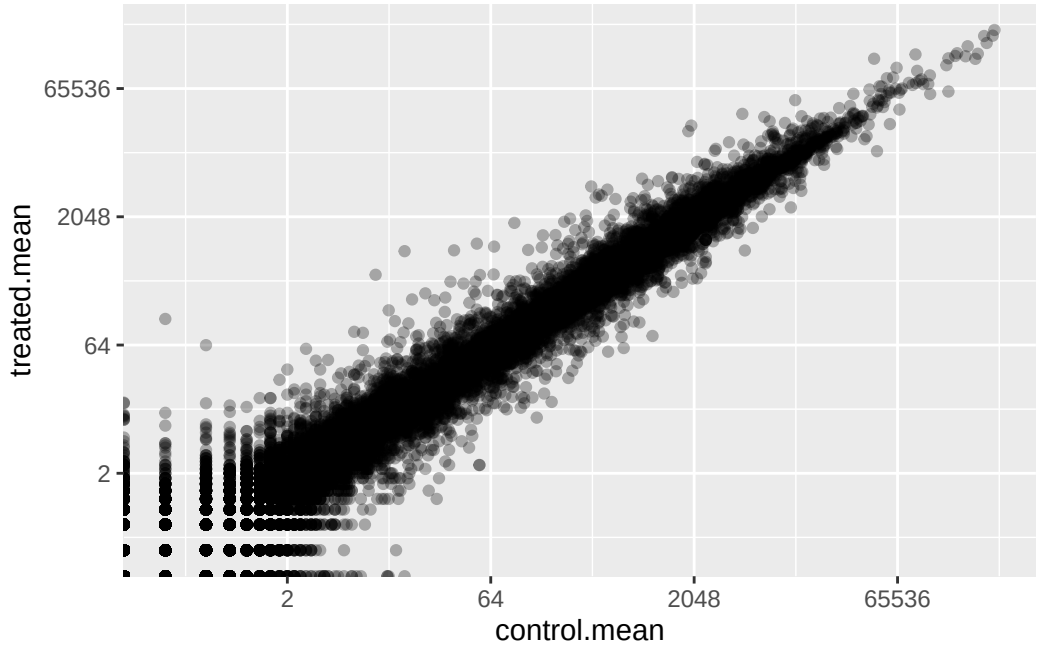
Q6. Try plotting both axes on a log scale.

The data is highly skewed, and is better visualized using log transformation.

```
ggplot(meancounts) +
  aes(control.mean, treated.mean) +
  geom_point(alpha=0.3) +
  scale_x_continuous(trans="log2") +
  scale_y_continuous(trans="log2")
```

```
Warning in scale_x_continuous(trans = "log2"): log-2 transformation introduced
infinite values.
```

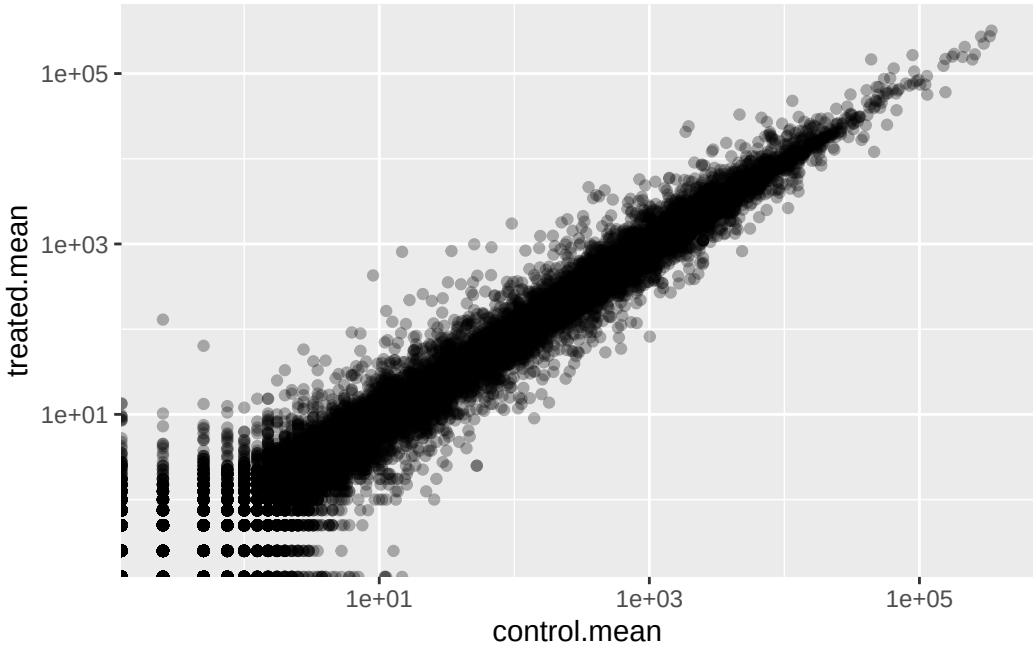
```
Warning in scale_y_continuous(trans = "log2"): log-2 transformation introduced
infinite values.
```




```
ggplot(meancounts) +
  aes(control.mean, treated.mean) +
  geom_point(alpha=0.3) +
  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.
```



Log2 units and fold change

If we consider treated/control counts, we will get a number that tells us the change.

```
# No change in the treated vs control
log2(20/20)
```

```
[1] 0
```

```
# A doubling in the treated vs control
log2(40/20)
```

```
[1] 1
```

```
log10(40/20)
```

```
[1] 0.30103
```

```
# A halving in the treated vs control
log2(10/20)
```

```
[1] -1
```

Q. Add a new column `log2fc` for log2 fold change of treated/control to our `meancounts` object.

```
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
```

Remove zero count genes

Typically, we would not consider zero count genes - as we have no data about them - and they should be excluded from further consideration. These lead to "funky" log2 fold change values (ex. divide by zero errors, etc.).

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

```
                row col
ENSG00000000005   2   1
ENSG00000004848  65   1
ENSG00000004948  70   1
ENSG00000005001  73   1
ENSG00000006059 121   1
ENSG00000006071 123   1
```

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

> 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=TRUE is needed to get an array of indexes of rows and columns with zeroes. We take the first column of the output to get the rows with zeroes, as those are the genes to be removed. We use unique() because one row number may appear multiple times if multiple columns in that row have zeroes.

```
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?

```
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?

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

```
[1] 367
```

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

No, because although this checks if there is a 2 fold change up or down, this does not mean the change is statistically significant.

DESeq analysis

We are missing and measure of signigicance from the work we have done so far. Let's do this properly with the **DESeq2** package.

```
library(DESeq2)
```

The DESeq2 package, like many Bioconductor packages, wants its input in a very specific way - a data structure setup with all the info it needs for the calculation.

```
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 main function from this package is called `DESeq()`. It will run full analysis for us on our `dds` input object:

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

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

Extract our results:

```
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
```

```
36000 * 0.05
```

```
[1] 1800
```

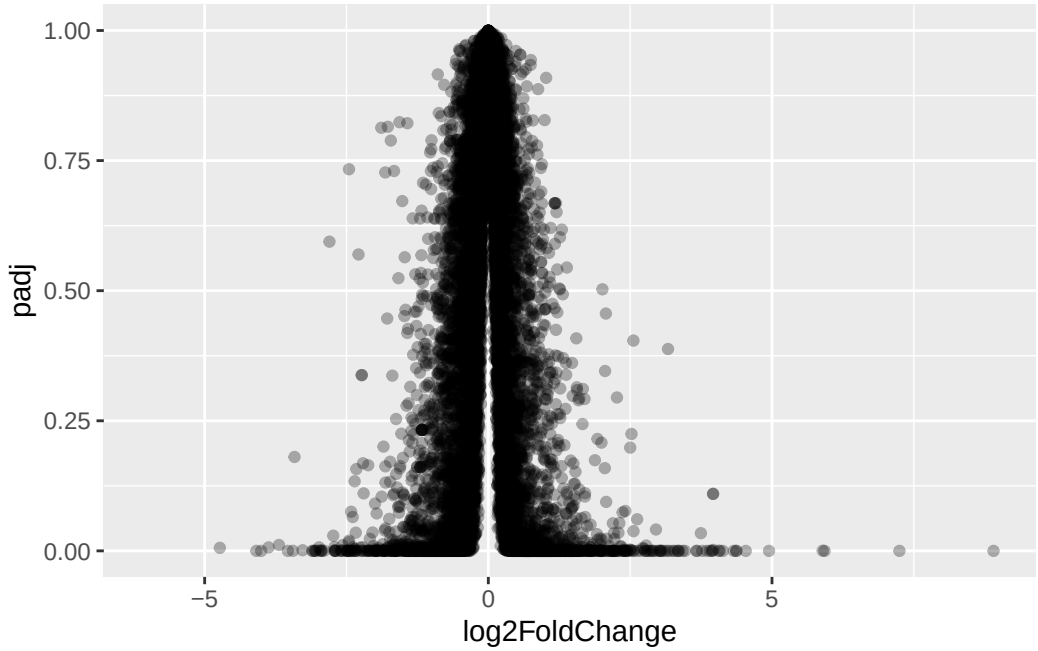
Volcano plot

A useful and ubiquitous summary figure of our results is often called a "volcano plot". It is basically a plot of log2 fold change values vs adjusted p-values.

Q. Use ggplot to make a first version "volcano plot" of `log2FoldChange` vs `padj`

```
ggplot(res) +
  aes(log2FoldChange, padj) +
  geom_point(alpha=0.3)
```

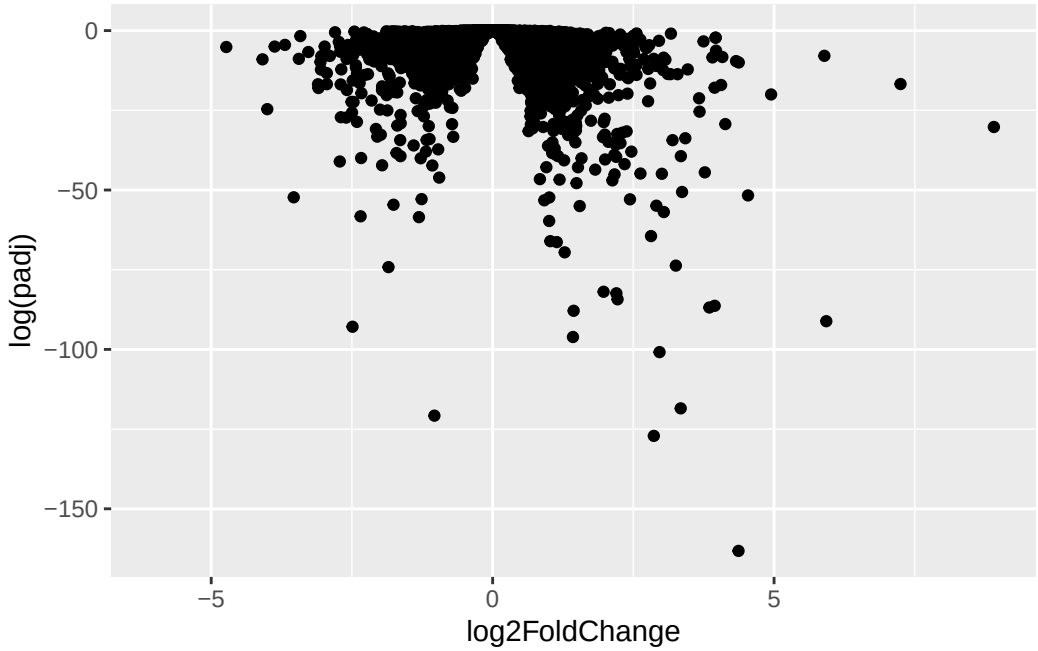
```
Warning: Removed 23549 rows containing missing values or values outside the scale range
(`geom_point()`).
```



This is not very useful because the y-axis (P-value) is not really helpful for seeing the low P-values (our focus).

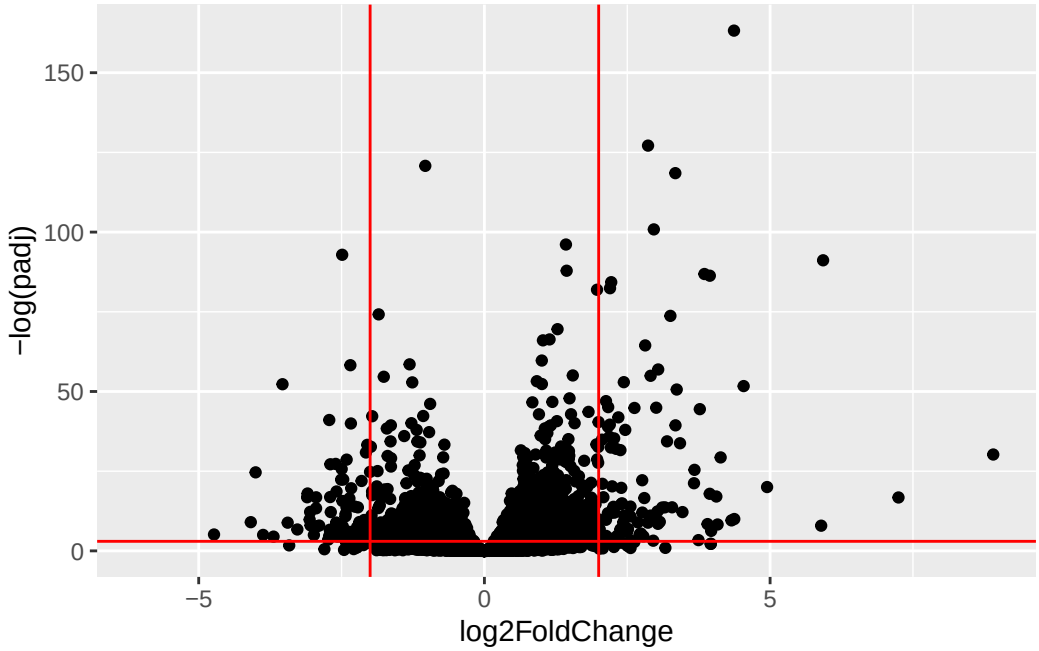
```
ggplot(res) +
  aes(log2FoldChange, log(padj)) +
  geom_point()
```

```
Warning: Removed 23549 rows containing missing values or values outside the scale range
(`geom_point()`).
```




```
ggplot(res) +
  aes(log2FoldChange, -log(padj)) +
  geom_point() +
  geom_vline(xintercept = c(-2,+2), col="red") +
  geom_hline(yintercept = -log(0.05), col="red")
```

```
Warning: Removed 23549 rows containing missing values or values outside the scale range
(`geom_point()`).
```



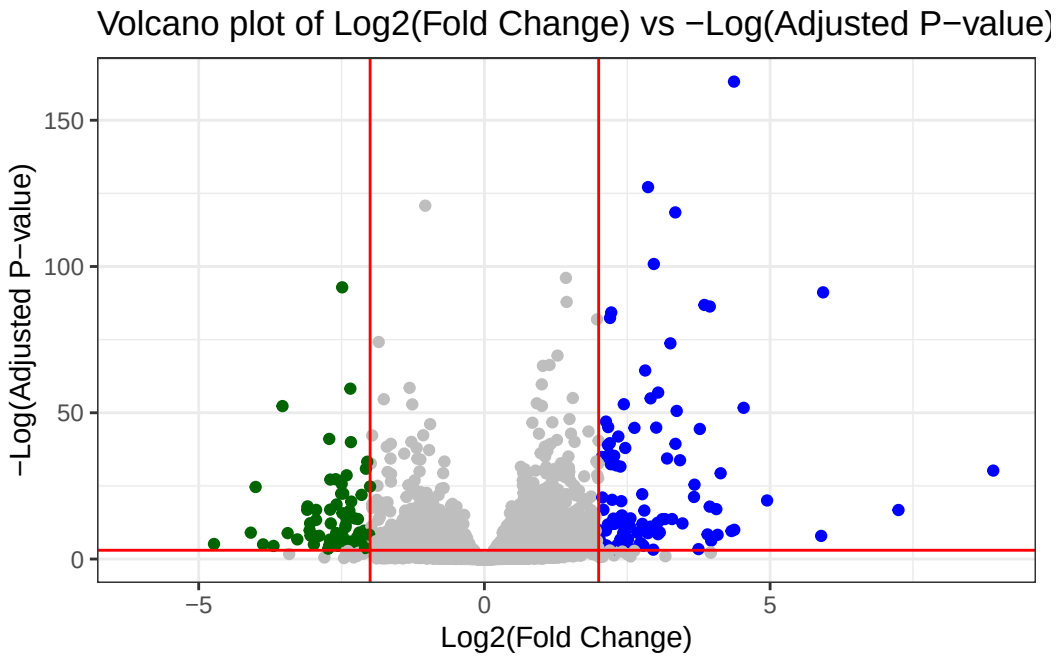
Add some plot annotation

Q. Add color to the points (genes) we care about, nice axis labels, a useful title, and a nice theme.

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

```
ggplot(res) +
 aes(log2FoldChange, -log(padj)) +
 geom_point(col = mycols) +
 geom_vline(xintercept = c(-2,+2), col="red") +
 geom_hline(yintercept = -log(0.05), col="red") +
 labs(x = "Log2(Fold Change)",
      y = "-Log(Adjusted P-value)",
      title = "Volcano plot of Log2(Fold Change) vs -Log(Adjusted P-value)") +
 theme_bw()
```

```
Warning: Removed 23549 rows containing missing values or values outside the scale range
(`geom_point()`).
```



Save our results to a CSV file

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

Add Annotation Data

To make sense of our results we need to know what the differentially expressed genes are and what biological pathways and processes they are involved in.

```
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
```

Let's start by mapping our ENSEMBLE ids to the more conventional gene SYMBOL.

We will use two Bioconductor packages for this "mapping", **AnnotationDbi** and **org.Hs.eg.db**

We will girst need to install these from Bioconductor with`BiocManager::install("")`

```
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"
```

```
res$symbol <- mapIds(org.Hs.eg.db,
                     keys = rownames(res), # Our ENSEMBL ids
                     keytype = "ENSEMBL",  # id format
                     column = "SYMBOL")    # translation output format
```

```
'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
```

Q. Can you add "GENENAME" and "ENTREZID" as new columns to **res**, named "name" and "entrez"?

```
res$name <- mapIds(org.Hs.eg.db,
                   keys = rownames(res), # Our ENSEMBL ids
                   keytype = "ENSEMBL",  # id format
                   column = "GENENAME")    # translation output format
```

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

```
res$entrez <- mapIds(org.Hs.eg.db,
                     keys = rownames(res), # Our ENSEMBL ids
                     keytype = "ENSEMBL",  # id format
                     column = "ENTREZID")    # translation output format
```

```
'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                    name      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
```

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

Pathway Analysis

Now that we know the gene names (gene symbols) and their entrez IDs, we can find out what pathway they are involed in. This is called "pathway analysis" or "gene set enrichment".

We will use **gage** package and then **pathview** packaged to do this analysis (but there are loads of others).

```
library(gage)
library(gageData)
library(pathview)
```

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

To run our pathway analysis we will use the **gage()** function. It wants two main inputs: a vector of importance (in our case: the log2 fold cahnge values) and the gene sets to check overlap for.

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

```
     TSPAN6        TNMD        DPM1       SCYL3       FIRRM         FGR
-0.35070302          NA  0.20610777  0.02452695 -0.14714205 -1.73228897
```

KEGG speaks entrez (i.e. uses ENTREZID format) not gene symbol format.

```
names(foldchanges) <- res$entrez
```

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

```
head(keggres$less, 5)
```

```
                                                    p.geomean stat.mean
hsa05332 Graft-versus-host disease                 0.0004250461 -3.473346
hsa04940 Type I diabetes mellitus                  0.0017820293 -3.002352
hsa05310 Asthma                                    0.0020045888 -3.009050
hsa04672 Intestinal immune network for IgA production 0.0060434515 -2.560547
hsa05330 Allograft rejection                       0.0073678825 -2.501419
                                                          p.val      q.val
hsa05332 Graft-versus-host disease                 0.0004250461 0.09053483
hsa04940 Type I diabetes mellitus                  0.0017820293 0.14232581
```

```
hsa05310 Asthma                                          0.0020045888 0.14232581
hsa04672 Intestinal immune network for IgA production 0.0060434515 0.31387180
hsa05330 Allograft rejection                             0.0073678825 0.31387180
                                                         set.size         exp1
hsa05332 Graft-versus-host disease                             40 0.0004250461
hsa04940 Type I diabetes mellitus                              42 0.0017820293
hsa05310 Asthma                                                29 0.0020045888
hsa04672 Intestinal immune network for IgA production          47 0.0060434515
hsa05330 Allograft rejection                                   36 0.0073678825
```

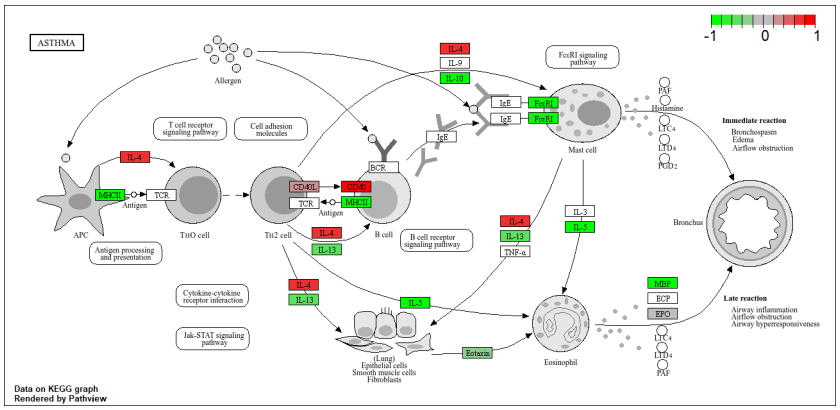
Let's make a figure of one of these pathways with our DEGs higlighted:

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

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

```
Info: Working in directory C:/Users/extra/Documents/school/BIMM 143/class13
```

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



Q. Generate and insert a pathway figure for "Graft-versus-hose disease" and "Type 1 diabetes".

```
pathview(foldchanges, pathway.id = "hsa05332") # graft-versus-host
```

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

```
Info: Working in directory C:/Users/extra/Documents/school/BIMM 143/class13
```

```
Info: Writing image file hsa05332.pathview.png
```

```
pathview(foldchanges, pathway.id = "hsa04940") # diabetes
```

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

Info: Working in directory C:/Users/extra/Documents/school/BIMM 143/class13

Info: Writing image file hsa04940.pathview.png
```