Class 12: RNAseq analysis

Jiayi Zhou (PID: A17856751)

Table of contents

Background

Today we will analyze some RNAseq data from Himes et at. on the effects of a common steroid (sexamethasone) on airway smooth muscle cells (ASM cells).

Are starting points is the "counts" data and "metadata" that contain the count values for eachj gene in their different experiments (i.e. cell lines with or without the drug).

Data import

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

Let's have a wee peak at these objects:

```
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
```

Q1.1. How many genes are in this dataset?

38,694 genes.

```
nrow(counts)
```

```
[1] 38694
```

Q1.2. How many different experiments (columns in counts(metadata) are there?

8 different experiments.

```
ncol(counts)
```

```
[1] 8
```

```
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
```

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

4 'control' cell lines.

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

```
[1] 4
```

Toy differential gene expression

To start our analysis let's calculate the mean counts for all genes in the "control experiments.

1. Extract all "control" columns from the `counts` object
2. Calculate the mean for all rows (i.e. genes) of these "control" columns 3-4. Do the same for "treated".
3. Compare these `control.mean` and `treated.mean` values.

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

To make the code more robust, use rowMeans() instead of dividing by a fixed number so the calculation automatically adjusts to any number of samples.

```
#Step 1. Extracted all the "control" columns
control.inds <- metadata$dex == "control"
control.counts <- counts[,control.inds]
head(counts[,control.inds])
```

```
                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
```

```
#Step 2, mean of all control columns.
control.means <- rowMeans(control.counts)
head(control.means)
```

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

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)

For the treated samples, select columns where `dex == "treated"` and calculate their per-gene averages using `rowMeans()` to create the `treated.means` vector.

```
#Step 3. Extracted all the "treated" columns
treated.inds <- metadata$dex == "treated"
treated.counts <- counts[,treated.inds]
head(counts[,treated.inds])
```

```
                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
```

```
#Step 4, mean of all treated columns.
treated.means <- rowMeans(treated.counts)
head(treated.means)
```

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

Store these together for ease of bookkeeping as `meancounts`

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

```
                control.means treated.means
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
```

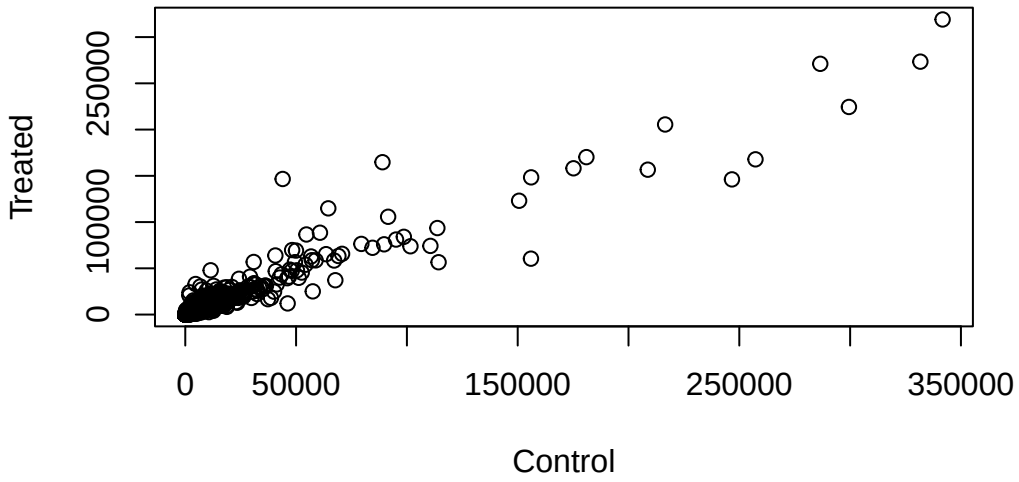
```
colSums(meancounts)
```

```
control.means treated.means
     23005324      22196524
```

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.

Make a plot of control vs treated mean values

```
plot(meancounts, xlab="Control", ylab="Treated")
```

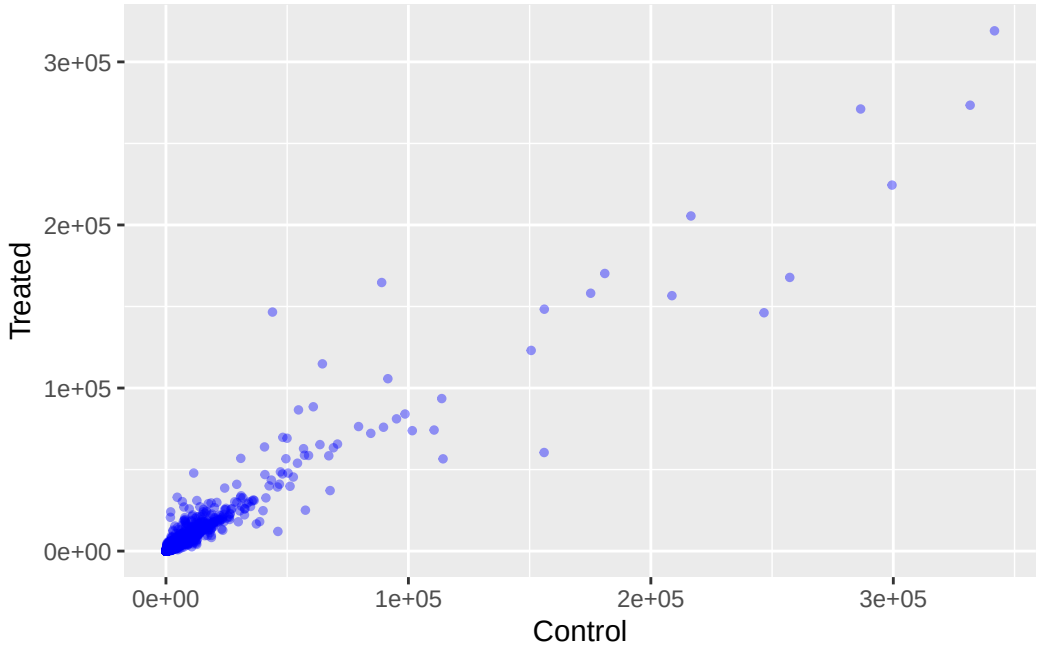


Q5 (b).You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot?

use the geom_point() function.

```
library(ggplot2)

ggplot(meancounts, aes(x = control.means, y = treated.means)) +
  geom_point(color = "blue", alpha = 0.4, size = 1) +
  labs(x = "Control", y = "Treated")
```



Make this a log log plot.

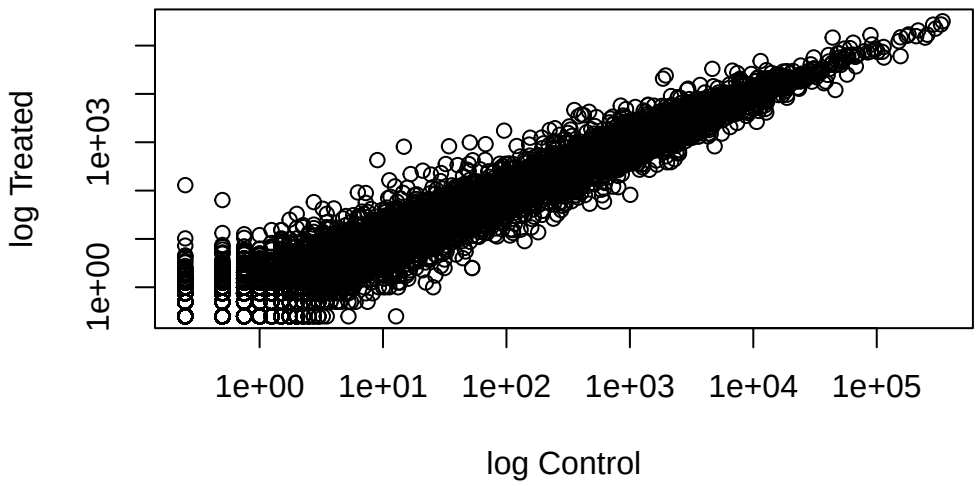
> Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

Use the `log = "xy"` argument in `plot()`.

```
plot(meancounts, log = "xy", xlab = "log Control", ylab = "log Treated")
```

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

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



We often talk metrics like "log2 fold-change"

```
# treated/control
# 0=no change, positive=up regulated, negative=down regulated
log2(10/10)
```

```
[1] 0
```

```
log2(10/20) # half the amount
```

```
[1] -1
```

```
log2(20/20) # double the amount
```

```
[1] 0
```

Let's calculate the log2 fold change for our treated over control mean counts.

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

```
                control.means treated.means      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
```

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

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

```
                control.means treated.means      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?

The `arr.ind = TRUE` argument makes `which()` return both row and column indices of zero values, and taking the first column with `unique()` ensures we only keep each gene's row index once even if it has multiple zeros.

A common "rule of thumb" is a log2 fold change cutoff of +2 and -2 to call genes "Up regulated" or "Down regulated".

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

The number of up-regulated genes is sum(up.ind) Number of "up" genes at +2 threshold

```
sum(meancounts$log2fc >= +2, na.rm=T)
```

```
[1] 1910
```

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

The number of down-regulated genes is sum(down.ind)

Number of "down" genes at -2 threshold

```
sum(meancounts$log2fc >= -2, na.rm=T)
```

```
[1] 23046
```

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

No, these results are based on fold change without normalization or statistical significance testing (e.g., DESeq2 p-values).

DESeq2 analysis

Let's do this analysis properly and keep our inner stats nerd happy - i.e. are the differences we see between drug and no drug significant given the replicate experiments.

```
library(DESeq2)
```

For DESeq analysis we need three things

- count values (`contData`)
- metadata telling us about the columns in `countdata` (coldata)
- design od the experiment (i.e.what do you want to compute)

Our first function from DESeq2 will set up the input required for analysis by storing all these 3 things together,

```
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 in DESeq2 that runs the analysis is caled `DESeq2`

```
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      -0.350703  0.168242 -2.084514 0.0371134
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160       0.206107  0.101042  2.039828 0.0413675
ENSG00000000457 322.664844       0.024527  0.145134  0.168996 0.8658000
ENSG00000000460  87.682625      -0.147143  0.256995 -0.572550 0.5669497
ENSG00000000938   0.319167      -1.732289  3.493601 -0.495846 0.6200029
                     padj
                <numeric>
ENSG00000000003  0.163017
ENSG00000000005        NA
ENSG00000000419  0.175937
ENSG00000000457  0.961682
ENSG00000000460  0.815805
ENSG00000000938        NA
```

```
summary(res)
```

```
out of 25258 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)       : 1564, 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
```

```
res05 <- results(dds, alpha=0.05)
summary(res05)
```

```
out of 25258 with nonzero total read count
adjusted p-value < 0.05
LFC > 0 (up)       : 1237, 4.9%
LFC < 0 (down)     : 933, 3.7%
outliers [1]       : 142, 0.56%
low counts [2]     : 9033, 36%
(mean count < 6)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

```
36000*0.05
```

```
[1] 1800
```

Adding Annotation Data

```
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=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>
ENSG00000000003 747.194195      -0.350703  0.168242 -2.084514 0.0371134
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160       0.206107  0.101042  2.039828 0.0413675
ENSG00000000457 322.664844       0.024527  0.145134  0.168996 0.8658000
ENSG00000000460  87.682625      -0.147143  0.256995 -0.572550 0.5669497
ENSG00000000938   0.319167      -1.732289  3.493601 -0.495846 0.6200029
                     padj      symbol
                <numeric> <character>
ENSG00000000003  0.163017      TSPAN6
ENSG00000000005        NA        TNMD
ENSG00000000419  0.175937        DPM1
ENSG00000000457  0.961682       SCYL3
ENSG00000000460  0.815805       FIRRM
ENSG00000000938        NA         FGR
```

Q11. Run the mapIds() function two more times to add the Entrez ID and UniProt accession and GENENAME as new columns called res$entrez$, res$uniprot and res$genename.

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

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

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

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

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

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

```
head(res)
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 10 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
                 <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195      -0.350703  0.168242 -2.084514 0.0371134
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160       0.206107  0.101042  2.039828 0.0413675
ENSG00000000457 322.664844       0.024527  0.145134  0.168996 0.8658000
ENSG00000000460  87.682625      -0.147143  0.256995 -0.572550 0.5669497
ENSG00000000938   0.319167      -1.732289  3.493601 -0.495846 0.6200029
                     padj      symbol      entrez     uniprot
                <numeric> <character> <character> <character>
```

```
ENSG00000000003  0.163017      TSPAN6        7105  A0A087WYV6
ENSG00000000005        NA        TNMD       64102      Q9H2S6
ENSG00000000419  0.175937        DPM1        8813      H0Y368
ENSG00000000457  0.961682       SCYL3       57147      X6RHX1
ENSG00000000460  0.815805       FIRRM       55732      A6NFP1
ENSG00000000938        NA         FGR        2268      B7Z6W7
                            genename
                         <character>
ENSG00000000003          tetraspanin 6
ENSG00000000005            tenomodulin
ENSG00000000419 dolichyl-phosphate m..
ENSG00000000457 SCY1 like pseudokina..
ENSG00000000460 FIGNL1 interacting r..
ENSG00000000938 FGR proto-oncogene, ..
```

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

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 10 columns
                 baseMean log2FoldChange     lfcSE      stat      pvalue
                <numeric>      <numeric> <numeric> <numeric>   <numeric>
ENSG00000152583   954.771        4.36836 0.2371306   18.4217 8.79214e-76
ENSG00000179094   743.253        2.86389 0.1755659   16.3123 8.06568e-60
ENSG00000116584  2277.913       -1.03470 0.0650826  -15.8983 6.51317e-57
ENSG00000189221  2383.754        3.34154 0.2124091   15.7316 9.17960e-56
ENSG00000120129  3440.704        2.96521 0.2036978   14.5569 5.27883e-48
ENSG00000148175 13493.920        1.42717 0.1003811   14.2175 7.13625e-46
                       padj      symbol      entrez     uniprot
                  <numeric> <character> <character> <character>
ENSG00000152583 1.33157e-71     SPARCL1        8404      B4E2Z0
ENSG00000179094 6.10774e-56        PER1        5187      A2I2P6
ENSG00000116584 3.28806e-53     ARHGEF2        9181  A0A8Q3SIN5
ENSG00000189221 3.47563e-52        MAOA        4128      B4DF46
ENSG00000120129 1.59896e-44       DUSP1        1843      B4DRR4
ENSG00000148175 1.80131e-42        STOM        2040      F8VSL7
                            genename
                         <character>
ENSG00000152583           SPARC like 1
ENSG00000179094 period circadian reg..
```

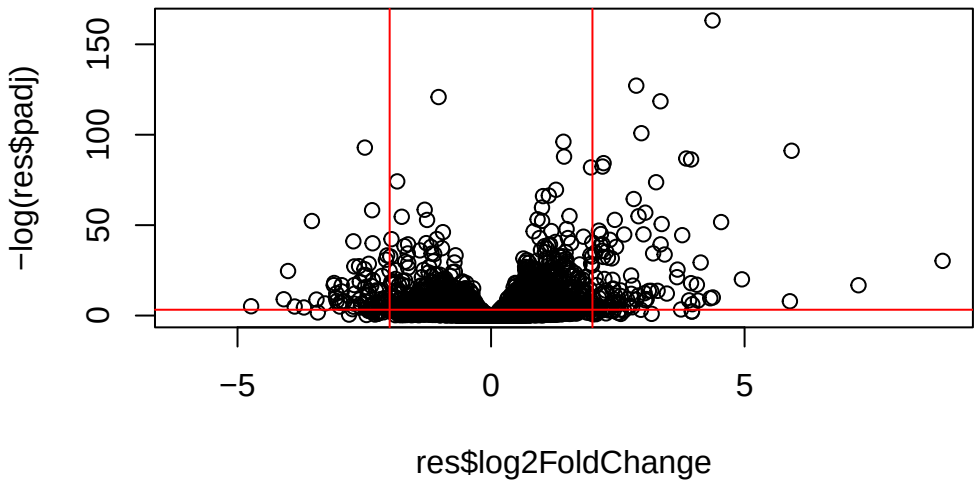
```
ENSG00000116584 Rho/Rac guanine nucl..
ENSG00000189221    monoamine oxidase A
ENSG00000120129 dual specificity pho..
ENSG00000148175               stomatin
```

```
write.csv(res[ord,], "deseq_results.csv")
```

Volcano Plot

This is a common summary result figure from these types of experiments and plot the log2 fold-change vs the adjusted p-value.

```
plot(res$log2FoldChange, -log(res$padj))
abline(v=c(-2,2), col="red")
abline(h=-log(0.04), col="red")
```




```
log(0.1)
```

```
[1] -2.302585
```

```
log(0.000001)
```

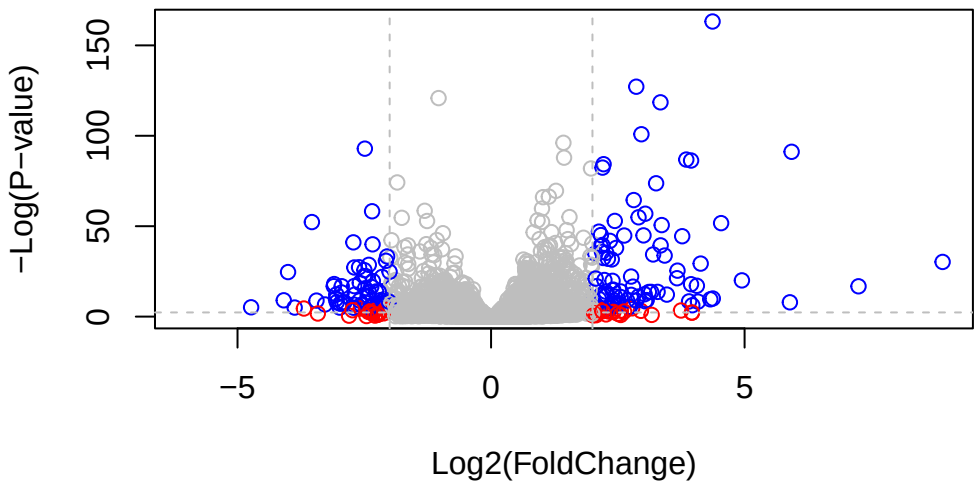
```
[1] -13.81551
```

```
# 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)
```




```
BiocManager::install("EnhancedVolcano")
```

```
Bioconductor version 3.22 (BiocManager 1.30.26), R 4.5.1 (2025-06-13)

Warning: package(s) not installed when version(s) same as or greater than current; use
  `force = TRUE` to re-install: 'EnhancedVolcano'
```

```r
library(EnhancedVolcano)
```

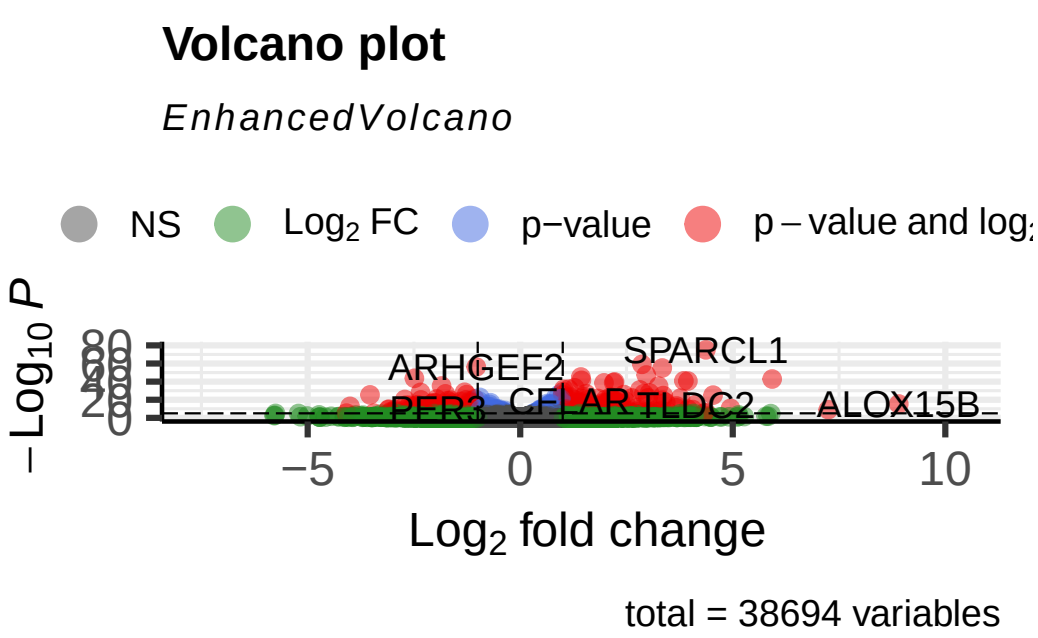
```
Loading required package: ggrepel
```

```r
x <- as.data.frame(res)

EnhancedVolcano(x,
    lab = x$symbol,
    x = 'log2FoldChange',
    y = 'pvalue')
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
i The deprecated feature was likely used in the EnhancedVolcano package.
  Please report the issue to the authors.

Warning: The `size` argument of `element_line()` is deprecated as of ggplot2 3.4.0.
i Please use the `linewidth` argument instead.
i The deprecated feature was likely used in the EnhancedVolcano package.
  Please report the issue to the authors.
```



Save our results

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