Lab 12: Transcriptomics and RNA-seq

Devanshi

2022-11-04

Table of contents

Installing packages

To start off, we need to install some Bioconductor packages to access the data. The code for this is:

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

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

Call the liibrary

```
#library(BiocManager)
library(DESeq2)
```

Importing the Count Data

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

```
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
```

Genes in this dataset:

```
dim(counts)
```

```
[1] 38694     8
```

```
dim(metadata)
```

```
[1] 8 4
```

There are 38694 genes in this dataset. There are 4 controls in this dataset.

Looking at Different gene expression

Considering `metadata`. We can find the sample IDs and mean counts per gene in these samples using the `dplyr` package.

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following object is masked from 'package:Biobase':

    combine

The following object is masked from 'package:matrixStats':

    count

The following objects are masked from 'package:GenomicRanges':

    intersect, setdiff, union

The following object is masked from 'package:GenomeInfoDb':

    intersect

The following objects are masked from 'package:IRanges':

    collapse, desc, intersect, setdiff, slice, union
```

```
The following objects are masked from 'package:S4Vectors':

    first, intersect, rename, setdiff, setequal, union

The following objects are masked from 'package:BiocGenerics':

    combine, intersect, setdiff, union

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

    filter, lag

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

    intersect, setdiff, setequal, union
```

```
control <- metadata %>% filter(dex=="control")
control.counts <- counts %>% select(control$id)
control.mean <- rowSums(control.counts)/4
head(control.mean)
```

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

This shows the mean counts per gene in the samples.

We can do the same for the treated samples.

```
treated <- metadata %>% filter(dex=="treated")
treated.counts <- counts %>% select(treated$id)
treated.mean <- rowSums(treated.counts)/4
head(treated.mean)
```

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

We can combine the mean counts for both the control and treated groups into a data frame.

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

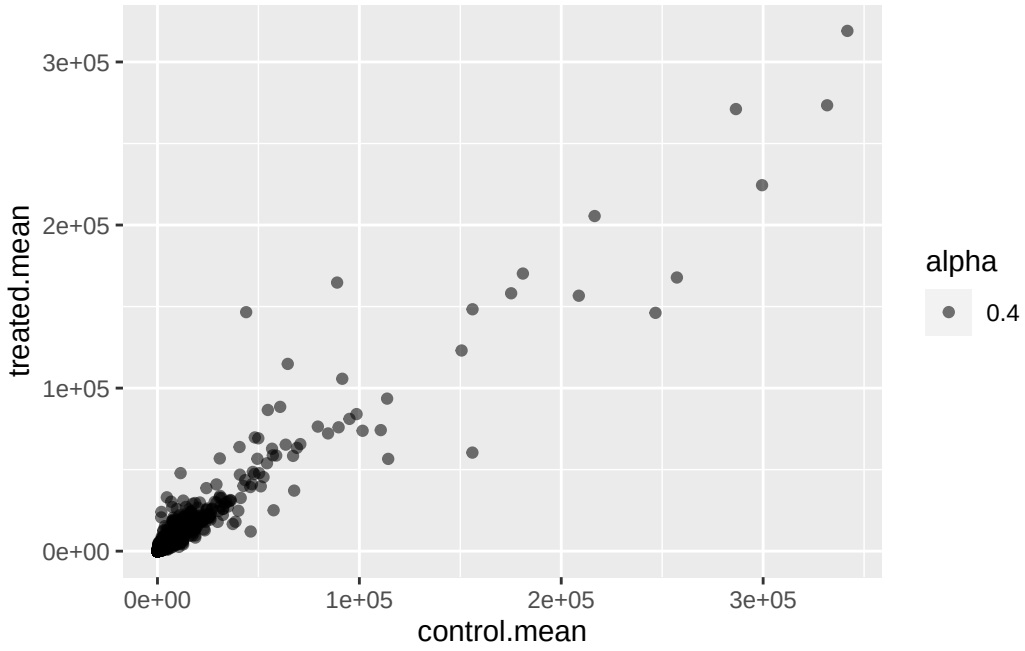
We can use `colSums()` to find the sums of the mean gene counts in each of the control and treated groups.

```
colSums(meancounts)
```

```
control.mean treated.mean
    23005324     22196524
```

We can also create a Scatter plot for this data frame to compare control and treated mean data.

```
library(ggplot2)
ggplot(meancounts) + aes(x=control.mean, y=treated.mean, alpha=0.4) + geom_point()
```

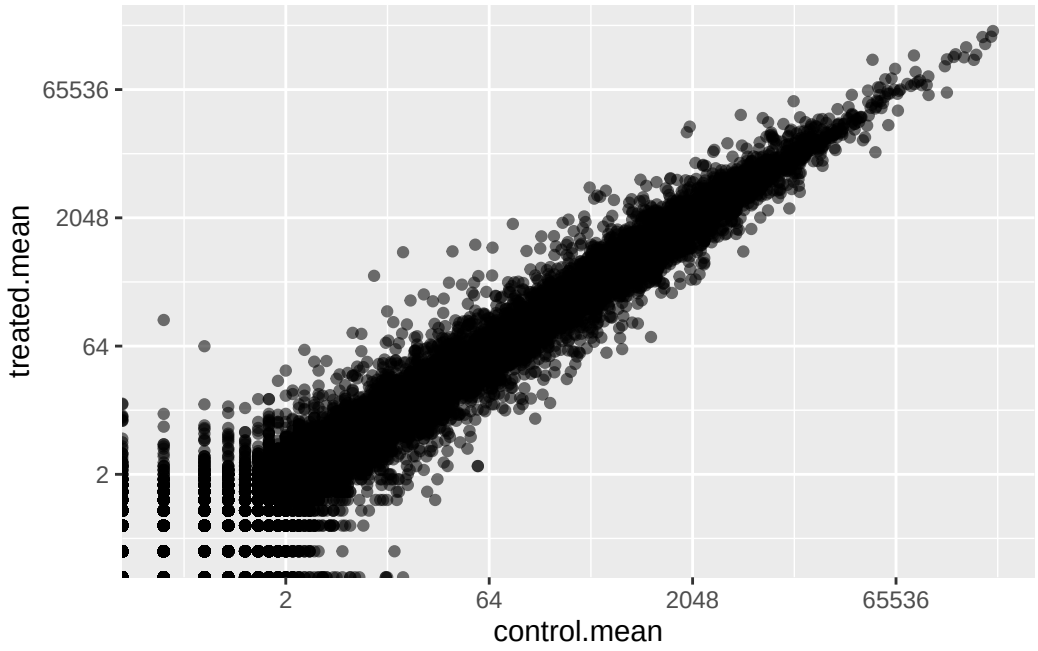


Making this on a logarithmic scale

```
ggplot(meancounts) + aes(x=control.mean, y=treated.mean, alpha=0.4) + geom_point(show.lege
```

```
Warning: Transformation introduced infinite values in continuous x-axis
```

```
Warning: Transformation introduced infinite values in continuous y-axis
```



It would be helpful to save these natural log results within the `meancounts` data set.

```
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
```

To get rid of the NaN and Inf results.

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

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
```

We have 21817 genes remaining

It's important to use the `arr.ind` argument within the which function to return the position indices which have zero values. It's important to use the `unique` function to avoid repeating rows and removing those values.

To check for differentially expressed genes, we can check which ones are >2 or <-2. We can find genes that are up or down regulated in this manner.

```
up.ind <- mycounts$log2fc > 2
down.ind <- mycounts$log2fc < (-2)
sum(up.ind==T)
```

```
[1] 250
```

```
sum(down.ind==T)
```

```
[1] 367
```

From these vectors, we can see that there are 250 upregulated and 367 downregulated genes with the parameters we previously defined. These results however are not very trustworthy because there is no statistical significance that has been performed on the log fold change values yet to draw any conclusions on whether these genes are truly up or down regulated.

Using DESeq 2 to look at gene data

This is a better method of looking at large data sets.

```
library(DESeq2)
citation("DESeq2")
```

```
To cite package 'DESeq2' in publications use:

  Love, M.I., Huber, W., Anders, S. Moderated estimation of fold change
  and dispersion for RNA-seq data with DESeq2 Genome Biology 15(12):550
  (2014)

A BibTeX entry for LaTeX users is

  @Article{,
    title = {Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2
    author = {Michael I. Love and Wolfgang Huber and Simon Anders},
    year = {2014},
    journal = {Genome Biology},
    doi = {10.1186/s13059-014-0550-8},
    volume = {15},
    issue = {12},
    pages = {550},
  }
```

```
# to see where this package came from
```

We can use the DESeq package directly to add our data, and organise by dex to differentiate between control and treated samples.

```
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
```

We can now run the DESeq function on our new data set, before displaying results from it.

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

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

We can now get results from these data.

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

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 38694 rows and 6 columns
                 baseMean log2FoldChange     lfcSE      stat    pvalue
                <numeric>      <numeric> <numeric> <numeric> <numeric>
```

```
ENSG00000000003  747.1942     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005    0.0000             NA        NA        NA        NA
ENSG00000000419  520.1342      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457  322.6648      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460   87.6826     -0.1471420  0.257007 -0.572521 0.5669691
...                    ...            ...       ...       ...       ...
ENSG00000283115  0.000000             NA        NA        NA        NA
ENSG00000283116  0.000000             NA        NA        NA        NA
ENSG00000283119  0.000000             NA        NA        NA        NA
ENSG00000283120  0.974916      -0.668258   1.69456 -0.394354  0.693319
ENSG00000283123  0.000000             NA        NA        NA        NA
                      padj
                 <numeric>
ENSG00000000003  0.163035
ENSG00000000005        NA
ENSG00000000419  0.176032
ENSG00000000457  0.961694
ENSG00000000460  0.815849
...                   ...
ENSG00000283115        NA
ENSG00000283116        NA
ENSG00000283119        NA
ENSG00000283120        NA
ENSG00000283123        NA
```

To view this data set:

```
res.df <- as.data.frame(res)
View(res.df)
```

Looking at a summary of the results:

```
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
```

We can change the p-value to be 0.05 by changing the `alpha` argument.

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

```
out of 25258 with nonzero total read count
adjusted p-value < 0.05
LFC > 0 (up)       : 1236, 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
```

Add Annotation Data

We can also add some gene names and annotations to our data to make it more meaningful using the `AnnotationsDbi` package. Make sure to install the BiocManager packages in the console.

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

The "org.Hs.eg.db" package shows genes from *Homo sapiens* across various databases.

```
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"
```

We can use the `mapIDs` function to add rows to our data results from above.

```
  # adding the ENSEMBL data column SYMBOL
  res$symbol <- mapIds(org.Hs.eg.db,
                       keys=row.names(res), # gene names
                       keytype="ENSEMBL",        # format of gene names
                       column="SYMBOL",          # new format
                       multiVals="first")
```

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

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

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

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

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

```
  # adding gene name
  res$genename <- mapIds(org.Hs.eg.db, keys = row.names(res), keytype = "ENSEMBL",
                         column = "GENENAME", 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     -0.3507030  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, ..
```

We can now order the results based on the p-value and view them.

```
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 0.2371268   18.4220 8.74490e-76
ENSG00000179094   743.253        2.86389 0.1755693   16.3120 8.10784e-60
ENSG00000116584  2277.913       -1.03470 0.0650984  -15.8944 6.92855e-57
ENSG00000189221  2383.754        3.34154 0.2124058   15.7319 9.14433e-56
ENSG00000120129  3440.704        2.96521 0.2036951   14.5571 5.26424e-48
ENSG00000148175 13493.920        1.42717 0.1003890   14.2164 7.25128e-46
                      padj      symbol      entrez     uniprot
                 <numeric> <character> <character> <character>
ENSG00000152583 1.32441e-71     SPARCL1        8404  A0A024RDE1
ENSG00000179094 6.13966e-56        PER1        5187      O15534
```

```
ENSG00000116584 3.49776e-53     ARHGEF2        9181      Q92974
ENSG00000189221 3.46227e-52        MAOA        4128      P21397
ENSG00000120129 1.59454e-44       DUSP1        1843      B4DU40
ENSG00000148175 1.83034e-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
```

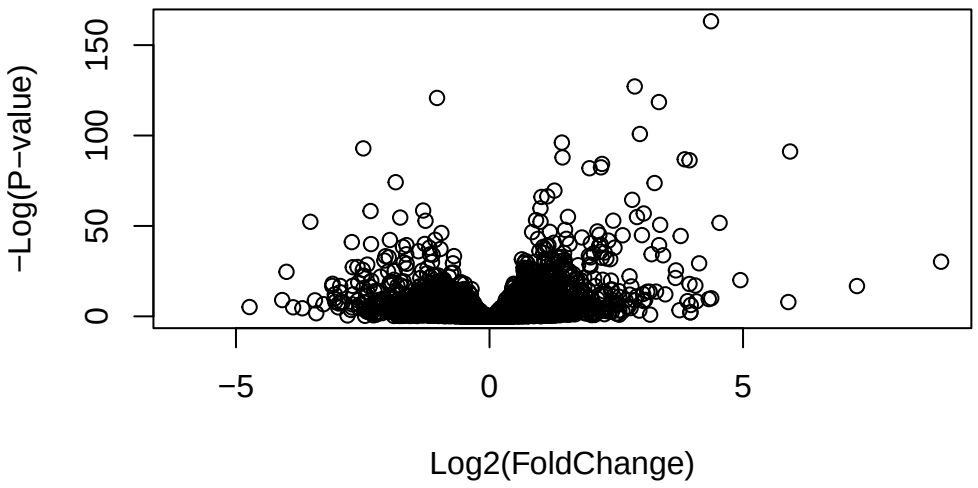
We can now save our ordered results in a csv file using the `write.csv` function.

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

Data Visualisation

We can visualise these data as a volcano plot.

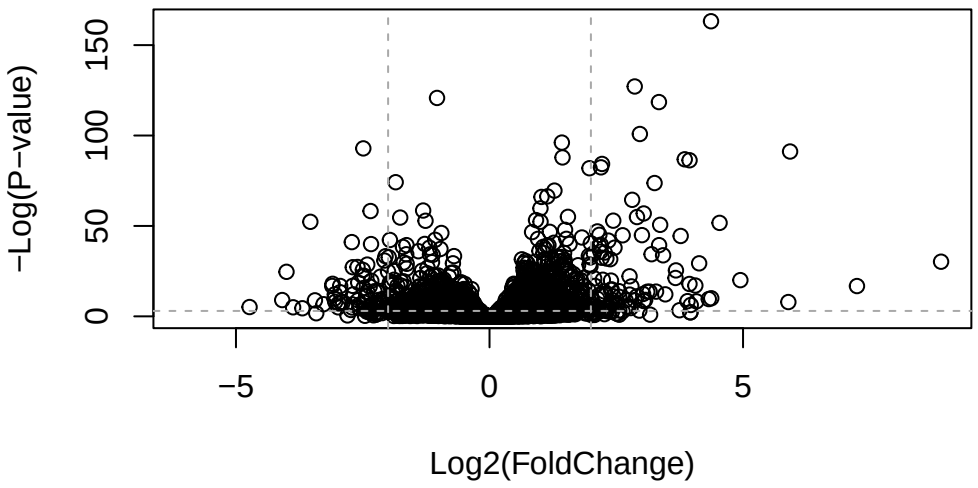
```
plot(res$log2FoldChange, -log(res$padj), xlab = "Log2(FoldChange)",
     ylab = "-Log(P-value)")
```



We can add some 'ablines' to make this a bit more meaningful in terms of our up and down regulation cut offs.

```
plot(res$log2FoldChange, -log(res$padj), xlab = "Log2(FoldChange)",
     ylab = "-Log(P-value)")

abline(v=c(-2,2), col="darkgray", lty=2)
abline(h=-log(0.05), col="darkgray", lty=2)
```



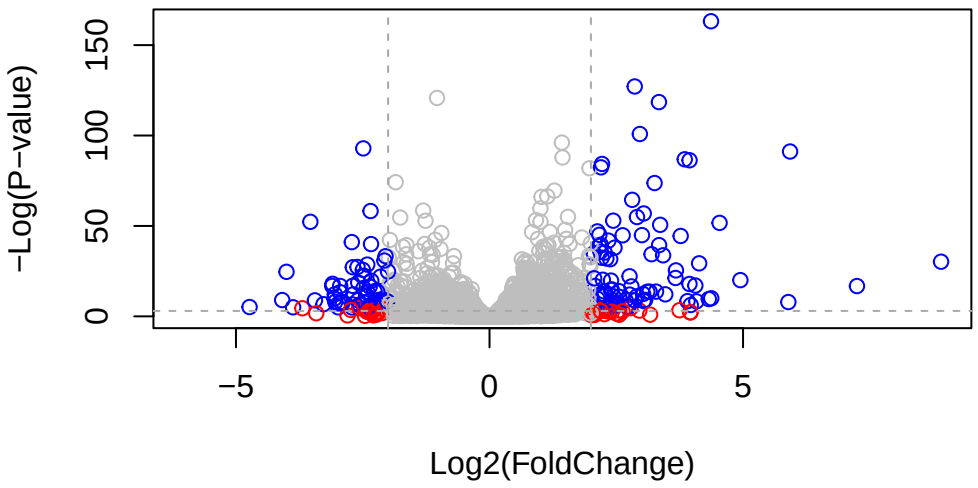
We can also add colors to this plot to make it more meaningful.

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

inds <- (res$padj<0.01) & (abs(res$log2FoldChange)>2)
mycols[inds] <- "blue"

plot(res$log2FoldChange, -log(res$padj), col = mycols,
     xlab = "Log2(FoldChange)", ylab = "-Log(P-value)")

abline(v=c(-2,2), col="darkgray", lty=2)
abline(h=-log(0.05), col="darkgray", lty=2)
```



Enhanced Volcano Plot

We can use the Enhanced Volcano Plot function to clean up the above volcano plot and add labels as well.

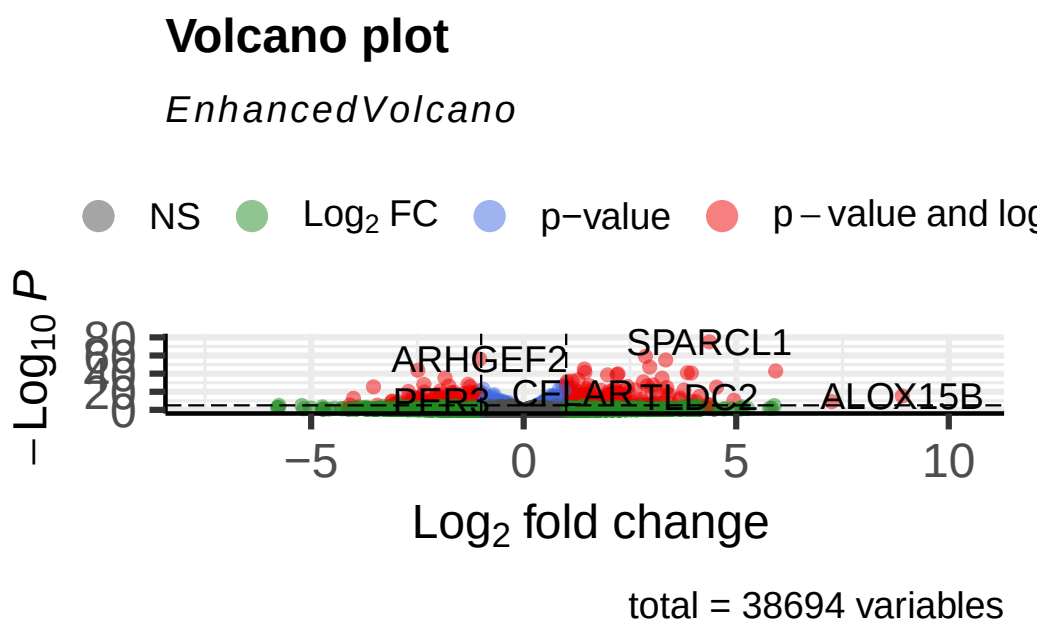
Make sure to install the BiocManager package for this function.

```
library(EnhancedVolcano)
```

```
Loading required package: ggrepel
```

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

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



This plot is a lot more meaningful in terms of looking at our data.

Pathway Analysis

Install the Biocmanager packages for this analysis function.

```
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)

# this contaings kegg pathways
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"
```

Now we can analyse our own data through this.

```
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
```