Class 12

Darby Patterson

Bioconductor and DESeq2 setup

```
library(BiocManager)
```

```
Bioconductor version '3.16' is out-of-date; the current release version '3.17'
  is available with R version '4.3'; see https://bioconductor.org/install
```

```
library(DESeq2)
```

```
Loading required package: S4Vectors
```

```
Loading required package: stats4
```

```
Loading required package: BiocGenerics
```

```

Attaching package: 'BiocGenerics'
```

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

    IQR, mad, sd, var, xtabs
```

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

    anyDuplicated, aperm, append, as.data.frame, basename, cbind,
    colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
    get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
```

```
    match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
    Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
    table, tapply, union, unique, unsplit, which.max, which.min


Attaching package: 'S4Vectors'

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

    expand.grid, I, unname

Loading required package: IRanges

Loading required package: GenomicRanges

Loading required package: GenomeInfoDb

Loading required package: SummarizedExperiment

Loading required package: MatrixGenerics

Loading required package: matrixStats


Attaching package: 'MatrixGenerics'

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

    colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
    colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
    colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
    colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
    colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
    colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
    colWeightedMeans, colWeightedMedians, colWeightedSds,
    colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
    rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
    rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
    rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
```

```
    rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
    rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
    rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
    rowWeightedSds, rowWeightedVars

Loading required package: Biobase

Welcome to Bioconductor

    Vignettes contain introductory material; view with
    'browseVignettes()'. To cite Bioconductor, see
    'citation("Biobase")', and for packages 'citation("pkgname")'.


Attaching package: 'Biobase'

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

    rowMedians

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

    anyMissing, rowMedians
```

Import countData and colData

```
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
```

Q1 How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

There are 38,694 genes in the dataset

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

```
control_cell = table(metadata$dex)['control']
control_cell
```

```
control
      4
```

We have 4 control cell lines

Toy differential gene expression

```
control <- metadata[metadata[,"dex"]=="control",]
control.counts <- counts[ ,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
```

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

Instead of doing rowSums(control.counts)/ 4 we could use rowMeans(control.count) so that it can apply to more than just this dataset.

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**)

```
treated <- metadata[metadata[,"dex"]=="treated",]
treated.mean <- rowSums( counts[ ,treated$id] )/4
names(treated.mean) <- counts$ensgene
head(treated.mean)
```

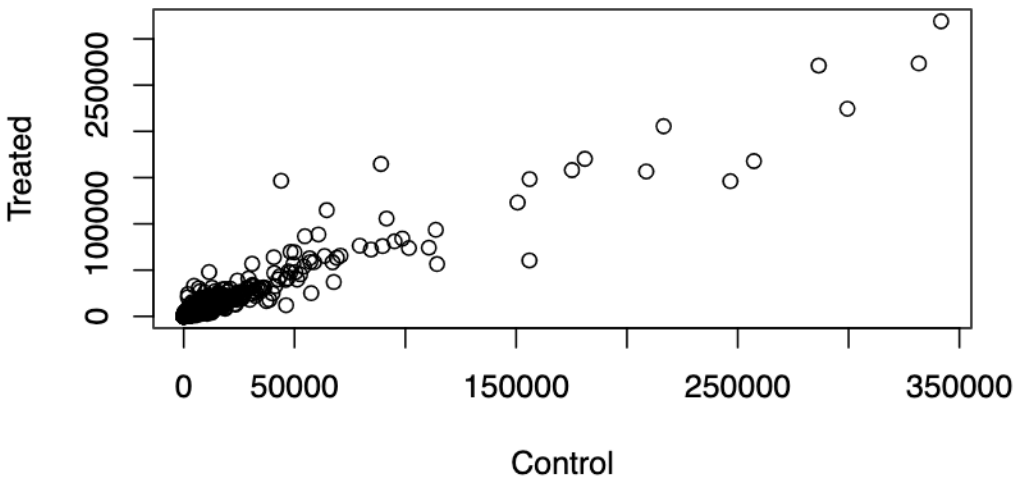
```
[1] 658.00   0.00 546.00 316.50  78.75   0.00
```

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

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

Q5 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.

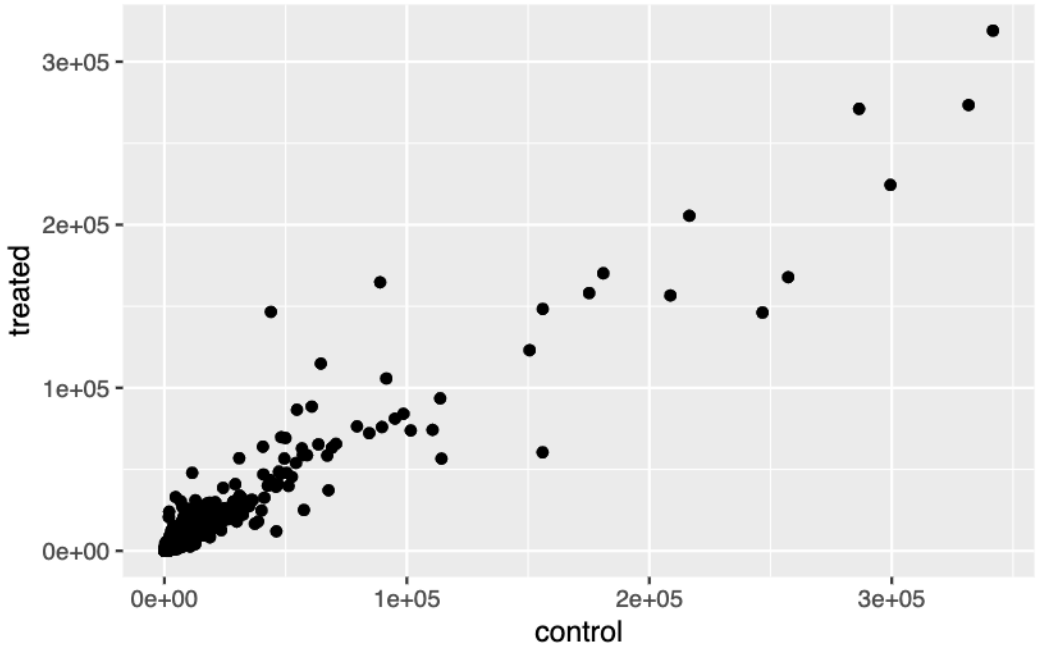
```
plot(meancounts[,1],meancounts[,2], xlab="Control", ylab="Treated")
```



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

geom_point

```
library(ggplot2)
ggplot(data = meancounts, aes(x = control.mean, y = treated.mean)) +
  geom_point() +
  xlab("control") +
  ylab("treated") +
  geom_point()
```

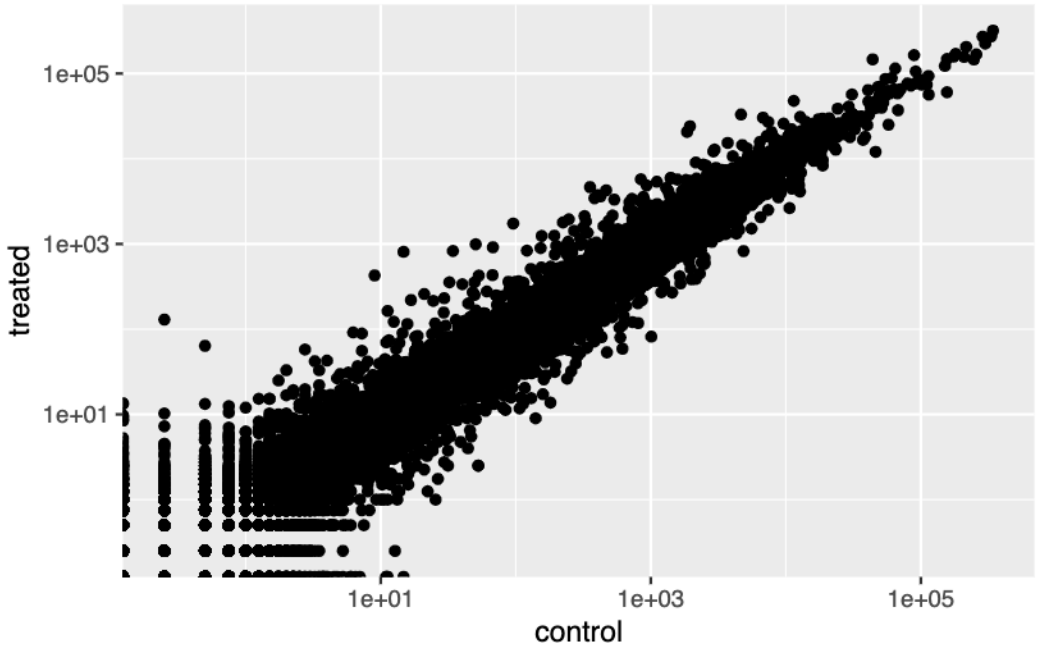


Q6 Try plotting both axes on a log scale. What is the argument to **plot()** that allows you to do this?

```
ggplot(data = meancounts, aes(x = control.mean, y = treated.mean)) +
  geom_point() +
  xlab("control") +
  ylab("treated") +
  scale_x_log10() +
  scale_y_log10()
```

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

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




```
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
```

We got some weird values of NaN and -Infinity, lets fix that.

```
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      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?

This function will show is what values in the genes and samples are 0, and unique() allows us to ensure we don't count a zero more than once.

Q8, 9 and 10: We also want to see which genes are up and down regulated. Do we trust these results?

```
up.ind <- mycounts$log2fc > 2
down.ind <- mycounts$log2fc < (-2)
up = sum(up.ind)
down = sum(down.ind)
cat("Up:", up, "\n")
```

```
Up: 250
```

```
cat("Down:", down, "\n")
```

```
Down: 367
```

We cannot call these results significant as we don't have enough information yet in our analysis. For now we can say that the data predicts more downregulated genes.

DESeq2 analysis

```
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},
  }
```

```
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
```

```
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)
#res
```

```
#as.data.frame(res)
```

```
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
```

```
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
```

Adding annotation data

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

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

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

```
#head(res)
```

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")

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

```
            #         multiVals="first")

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

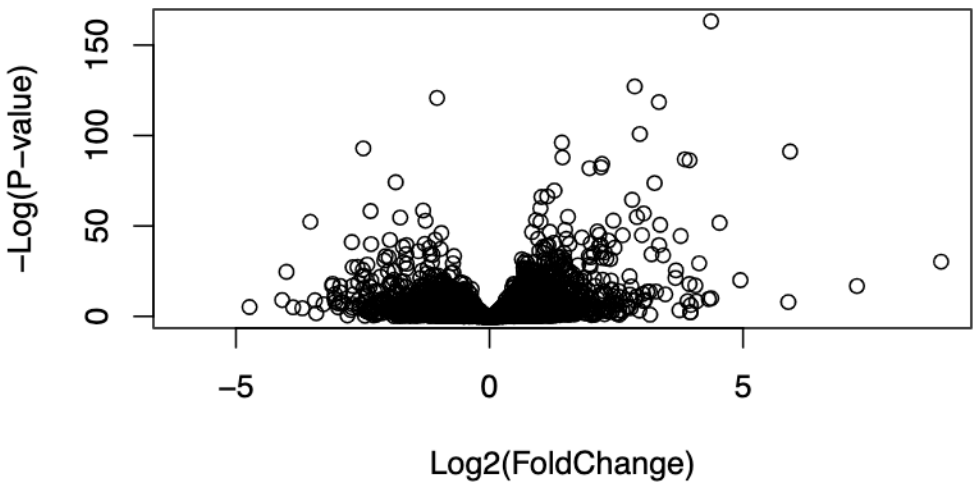
```
#ord <- order( res$padj )
#View(res[ord,])
#head(res[ord,])
```

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

Data Visualization

Volcano Plots

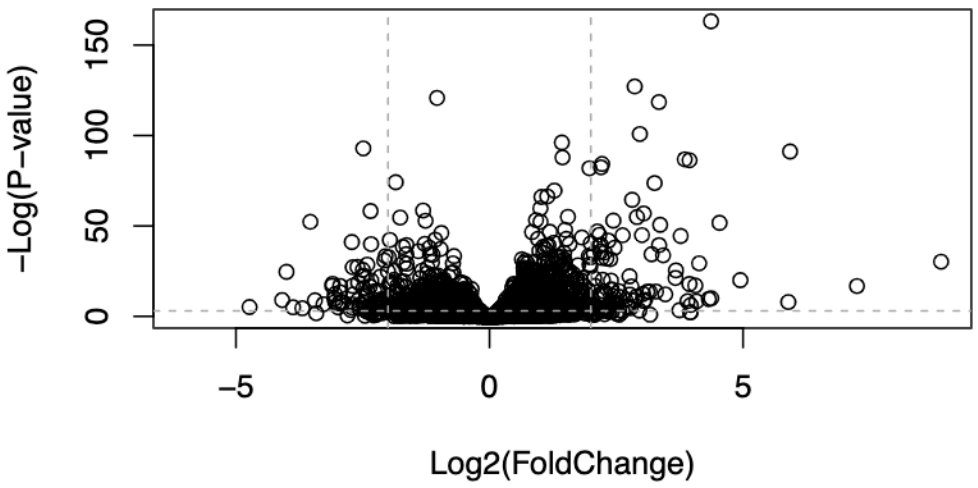
```
plot( res$log2FoldChange,  -log(res$padj),
      xlab="Log2(FoldChange)",
      ylab="-Log(P-value)")
```



We can add labels

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

# Add some cut-off lines
abline(v=c(-2,2), col="darkgray", lty=2)
abline(h=-log(0.05), col="darkgray", lty=2)
```



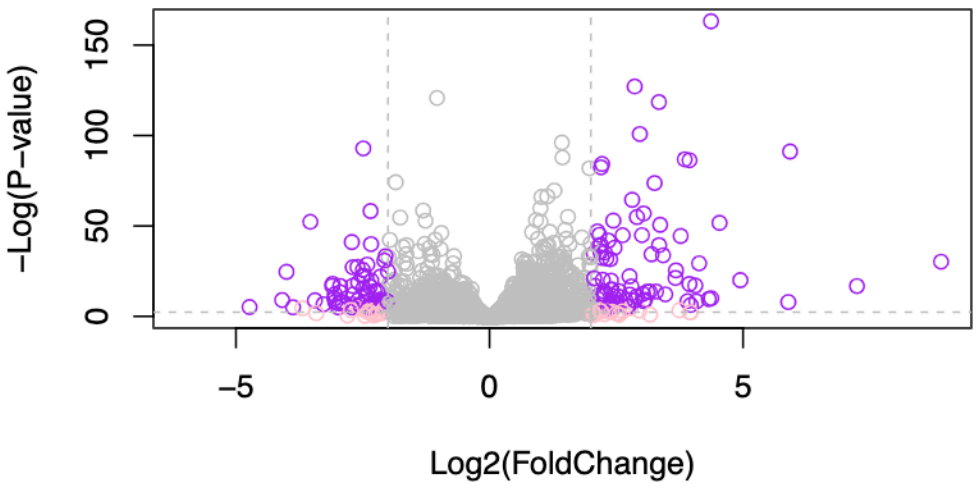
Now setting up a color vector

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

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

# 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)
```

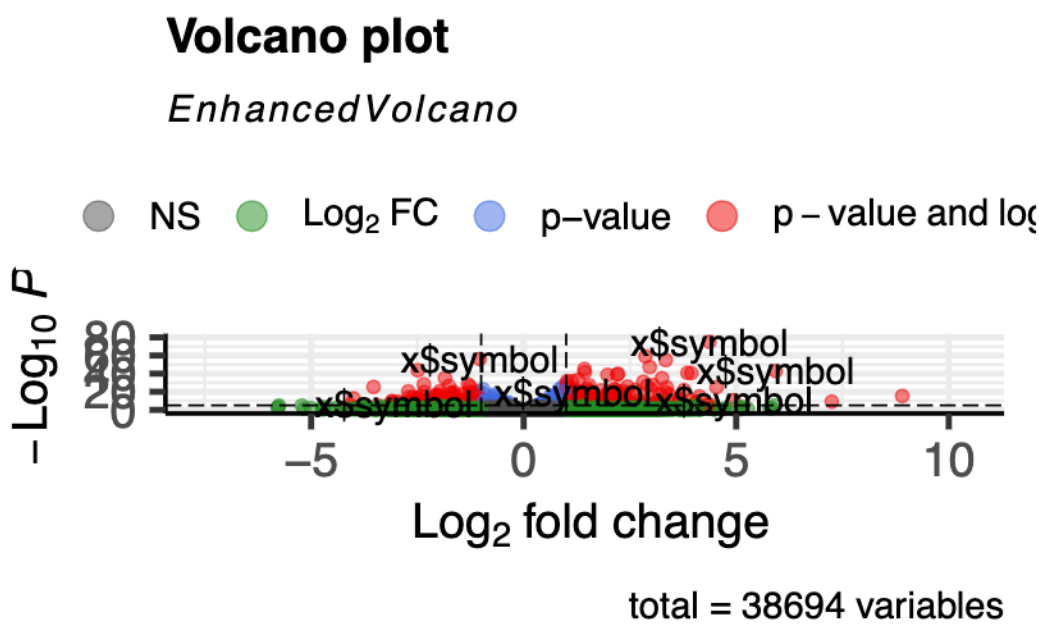



```
#BiocManager::install("EnhancedVolcano")
library(EnhancedVolcano)
```

```
Loading required package: ggrepel
```

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

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


Volcano plot
EnhancedVolcano
NS
Log2 FC
p–value
p – value and log
– Log10 P
x$symbol
x$symbol
x$symbol
x$symbol
x$symbol
x$symbol
–5
0
5
10
Log2 fold change
total = 38694 variables