Class13:Transcriptomics_And_RNA-Analysis

Isabel Philip (A16855684)

Today we will analyze the data from a published RNA-seq experiment where airway smooth muscle cells were treated with dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al. 2014).

```
library(BiocManager)
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, saveRDS, setdiff,
    table, tapply, union, unique, unsplit, which.max, which.min
```

```
Attaching package: 'S4Vectors'

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

    findMatches

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
```

1. Import countData and colData

There are 2 datasets I need to import/ read/ - `countData` the transcript counts per gene (rows) in the different experiements. - `colData` infomration about the columns (i.e. experiments) in `countData`.

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

Can have a sneak peak with `head()`

```
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?

There are 38694 genes in this data set.

```
nrow(counts)
```

```
[1] 38694
```

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

There are 4 "control" cell lines.

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

```
[1] 4
```

2. Toy differential gene expression

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

OR

```
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)
```

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

rowSums

We can find the average (mean) count values per gene for all the "control" experiments and compare it to the mean values of the "treated".

- Extract all "control" columns/ experiments from the `counts` data.
- Find the mean value for each gene in these columns

```
control.inds <- metadata$dex == "control"
control.count <- counts[ ,control.inds]
#will give 4 columns from the original data
```

```
dim(control.counts)
```

```
[1] 38694     4
```

> Mean from the summary of the columns (mean across the rows) - making into one column

Now find the row wise mean

```
control.mean <- rowSums(control.counts)/4
#dividing by 4 since there are 4 control groups
head(control.mean)
```

```
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`)

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

OR

```
treated.inds <- metadata$dex == "treated"
treated.count <- counts[ ,treated.inds]
treated.mean <- rowSums(treated.counts)/ncol(treated.counts)
head(treated.mean)
```

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

Have hardcorded "4" - another way to write that will account for the number of control groups, regardless of number

```
control.mean <- rowSums(control.counts)/ncol(control.counts)
head(control.mean)
```

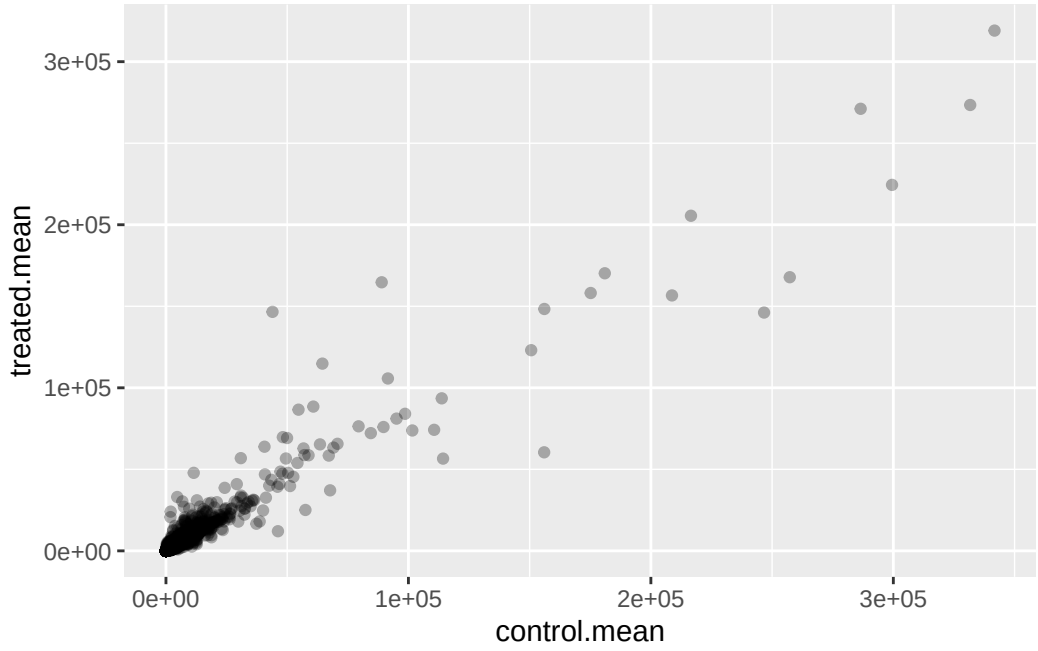
```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         900.75            0.00          520.50          339.75           97.25
ENSG00000000938
           0.75
```

For book keeping purposes, will combine the meancount data

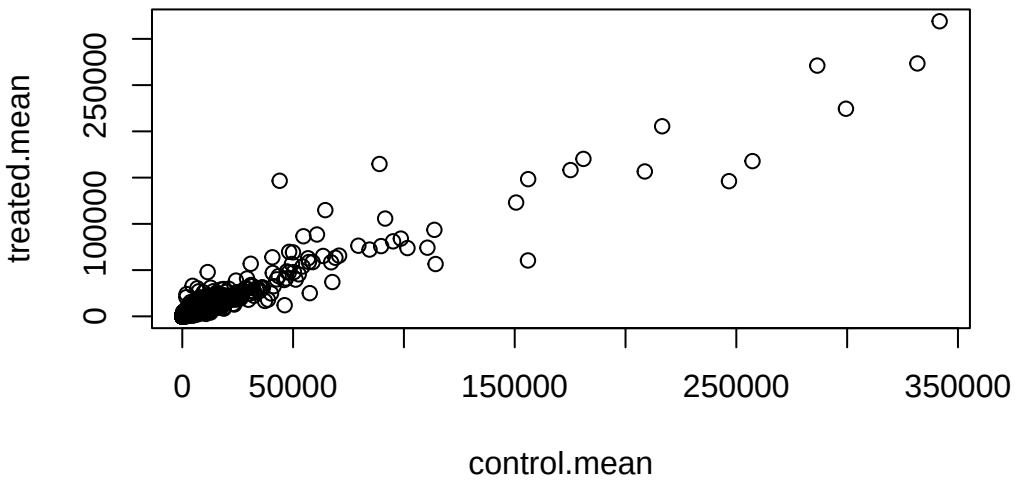
```
meancounts <- data.frame(control.mean, treated.mean)
#will show two columns
```

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.

```
library(ggplot2)
ggplot(meancounts) +
  aes(control.mean, treated.mean)+
  geom_point(alpha=0.3)
```




```
plot(meancounts)
```



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?

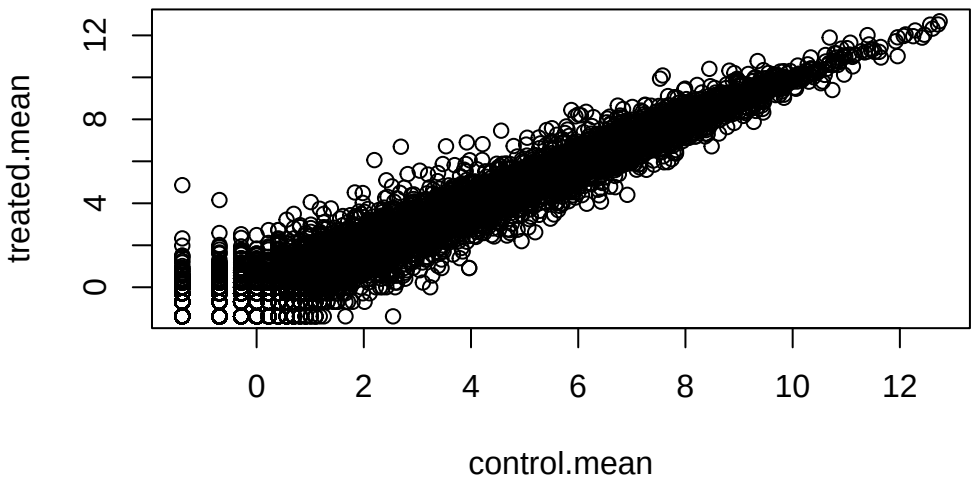
Geom_point()

Whenever we see data that is so heavily skewed, like this, we often transform it to log to see what is going on more easily.

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

log

```
plot(log(meancounts))
```



We most often work in log2 units as this makes math easier. Let's have a play to see this.

```
# treated / control
log2(20/20)
```

```
[1] 0
```

```
#log2 of 1 is 0, therefore if there is a log2 change of 0 - there was no change
```

```
log2(40/20)
```

```
[1] 1
```

```
# going to have a log2 change of 1
```

```
log2(80/20)
```

```
[1] 2
```

```
# going to have a log 2 change of 2
```

```
#treated/ control
log2(20/40)
```

```
[1] -1
```

```
# half as much treated as there is control, going into negatives
# sign of the value tells which direction
# 0 = on the line
# positive = above the line
# negative = below the line
```

Let's add "log2 fold-change" to our `meancounts` dataset.

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

There are a couple of "weird" results. Namely, the NaN ("not a number") and -Inf (negative infinity) results. The NaN is returned when you divide by zero and try to take the log. The -Inf is returned when you try to take the log of zero. It turns out that there are a lot of genes with zero expression.

Let's filter out zero count genes - i.e. remove the rows (genes) that have a 0 value in either control or treated means.

How many genes are "up" regulated at the common log2 fold change threshold of 2+?

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

```
[1] 1910
```

How many genes are "down" regulated at the threshold of -2?

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

```
[1] 23046
```

Q7. What is the purpose of the arr.ind argument in the which() function call below? Why would we then take the first column of the output and need to call the unique() function?

The purpose of the `arr.ind` argument in the which() function is being called is to return the row and column positions where it is `TRUE`. This will tell which genes (rows) and samples (columns) have 0 counts. The goal is to remove/ ignore any genes with 0 counts in any of the samples so we can just focus on the row answer. The `unique()` is being called to ensure that no row is being counted twice if it has 0 entries in both samples.

```
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
```

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

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?

367

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

```
[1] 250
```

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

```
[1] 367
```

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

These results cannot be fully trusted as we did not filter out significant from non-significant results. Fold-change can pear large without statistical significance, and therefore affect the results. Since we haven't looked at significance, the results can be misleading.

Setting up for DESeq/ DESeq Analysis

To do this the right way, we need to consider statistical significance of the differences - not just their magnitude.

```
#/ message: false
library(DESeq2)
```

To use this package, it wants `countData` and `colData` in a specific format

```
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
```

Extract my results

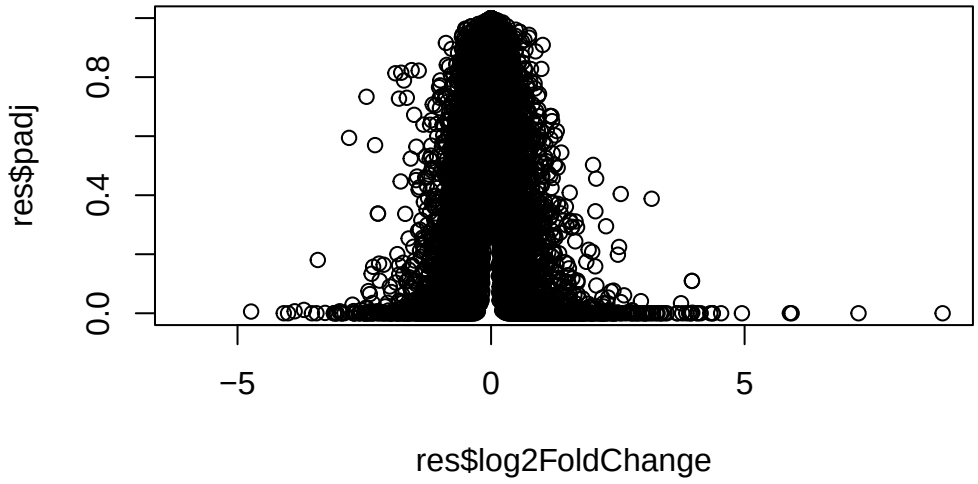
```
res <- results(dds)
head(res)
```

```
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
```

padj = p value adjusted

Plot of Fold Change vs. P-Value (adjusted for multiple testing)

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

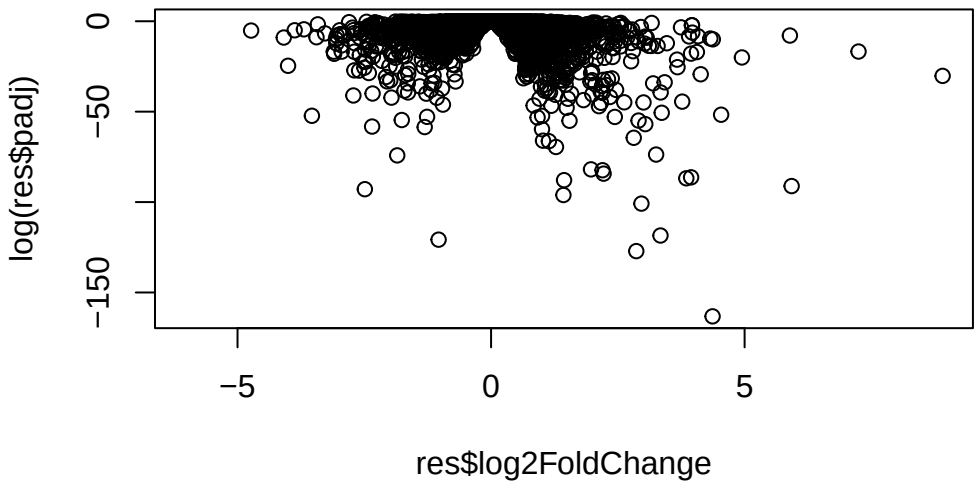


fold change of 0 = no change, negative = downregulated, positive = up regulated

the higher the p-value = the less significant it is WE WANT SMALL P-VALUES (downwards values)

Take the log of the P-Value

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



In this plot, look down the axis.

```
log(0.01)
```

```
[1] -4.60517
```

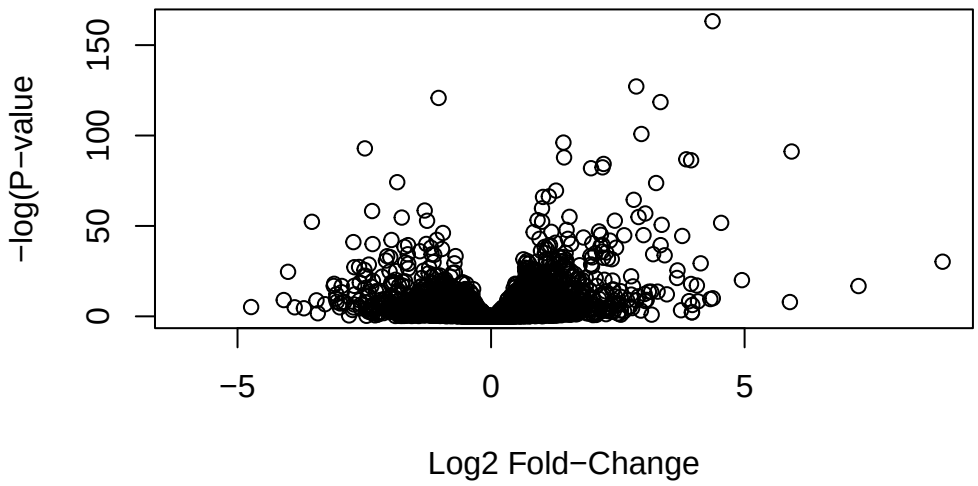
```
log(0.00000001)
```

```
[1] -18.42068
```

the smaller the p-value, the higher the negative number = greater significance

We can flip the y-axis by putting a minus sign on it

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



standard volcano plot

Let's save our work to date

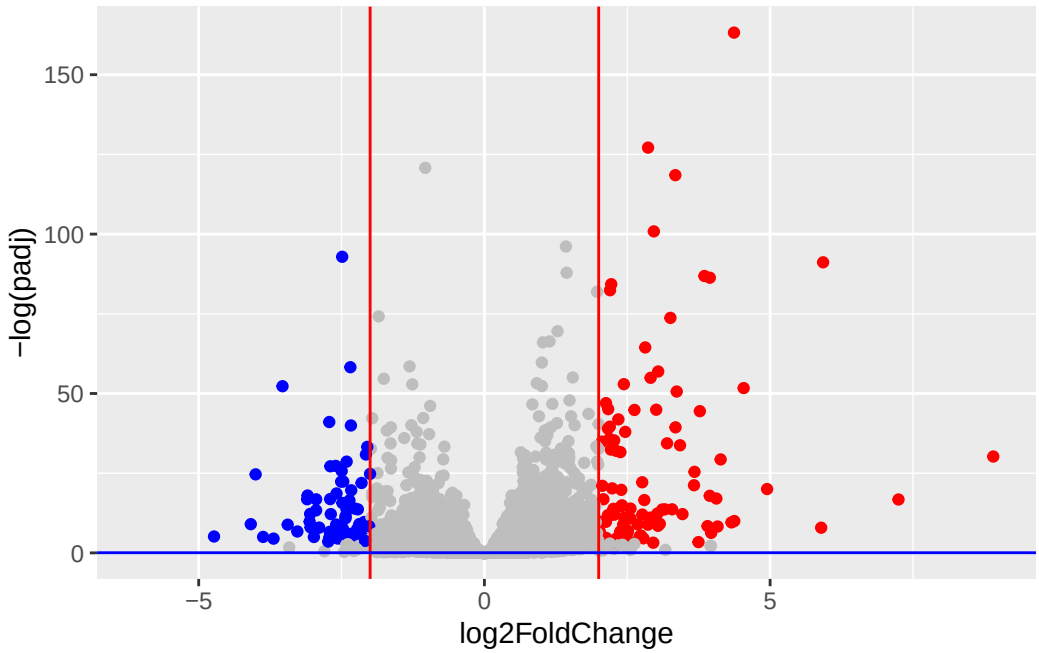
```
write.csv(res, file = "myresults.csv")
```

To finish off, let's make a nicer volcano plot. Use ggplot. - Add the log2 threshold lines of +2/-2 - Add P-value threshold lines at 0.05, - Add color to highlight the subset of genes that meet both of the above thresholds.

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

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


150
100
50
0
−log(padj)
−5
0
5
log2FoldChange