

# class15

Nicole Manuguid

11/16/2021

## Background

Our data for today came from Himes. et al. RNASeq analysis of the drug dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al. 2014).

Read the countData and colData

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

Let's have a look at these

```
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
```

```
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. 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
```

First I need to extract all the “control” columns. Then I will take the rowwise mean to get the average count values for all genes in these four experiments.

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

Avoid dividing by 4 because in the future more data could be added to this dataset.

```
control inds <- metadata$dex == "control"  
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
```

```
control.mean <- rowMeans(control.counts)  
# control.mean
```

Now do the same for the drug treated experiments (ie. columns)

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

```
##          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
```

```
treated.mean <- rowMeans(treated.counts)
```

We will combine our meancount data for bookkeeping purposes.

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

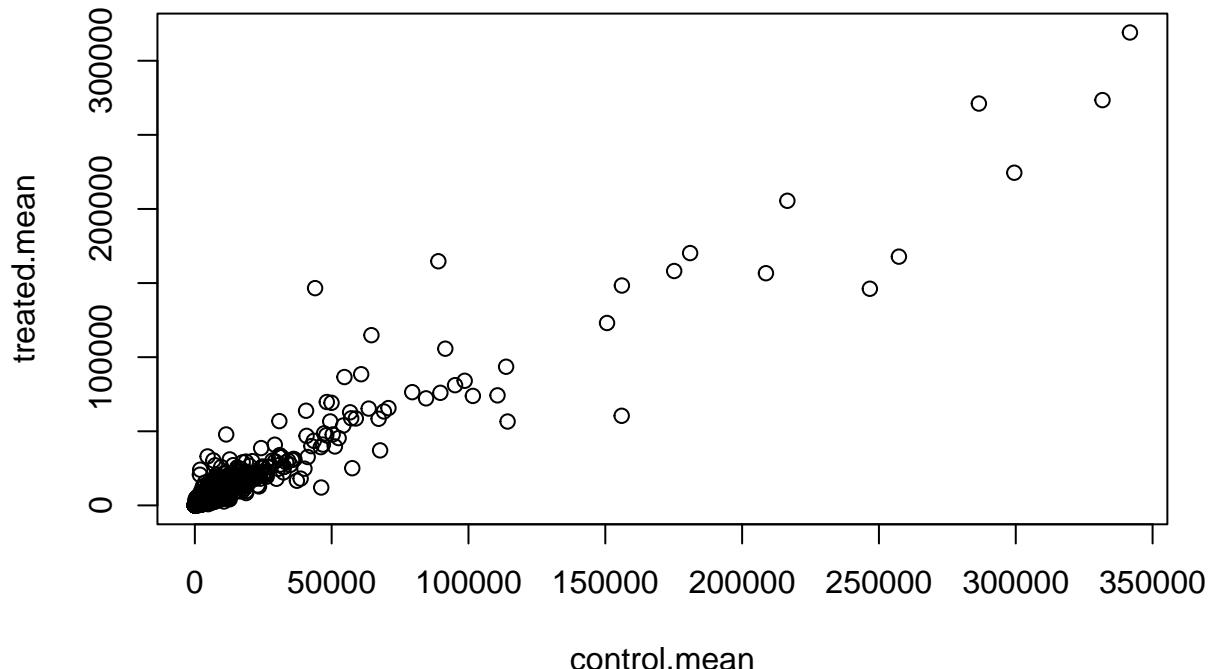
colSums(meancounts)

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

Let's make a quick plot

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.

```
plot(meancounts)
```



This screams at us that we need a log transformation to see details of our data!

I am going to re-plot on a log scale

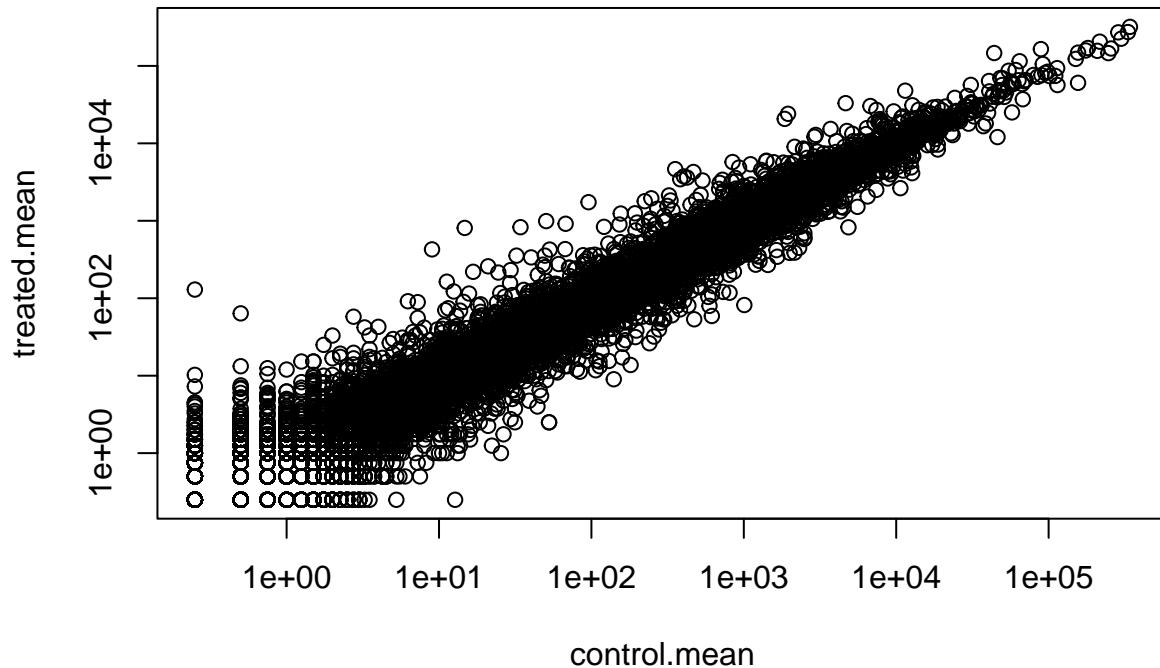
```

plot(meancounts, log = "xy")

## 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

```



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?

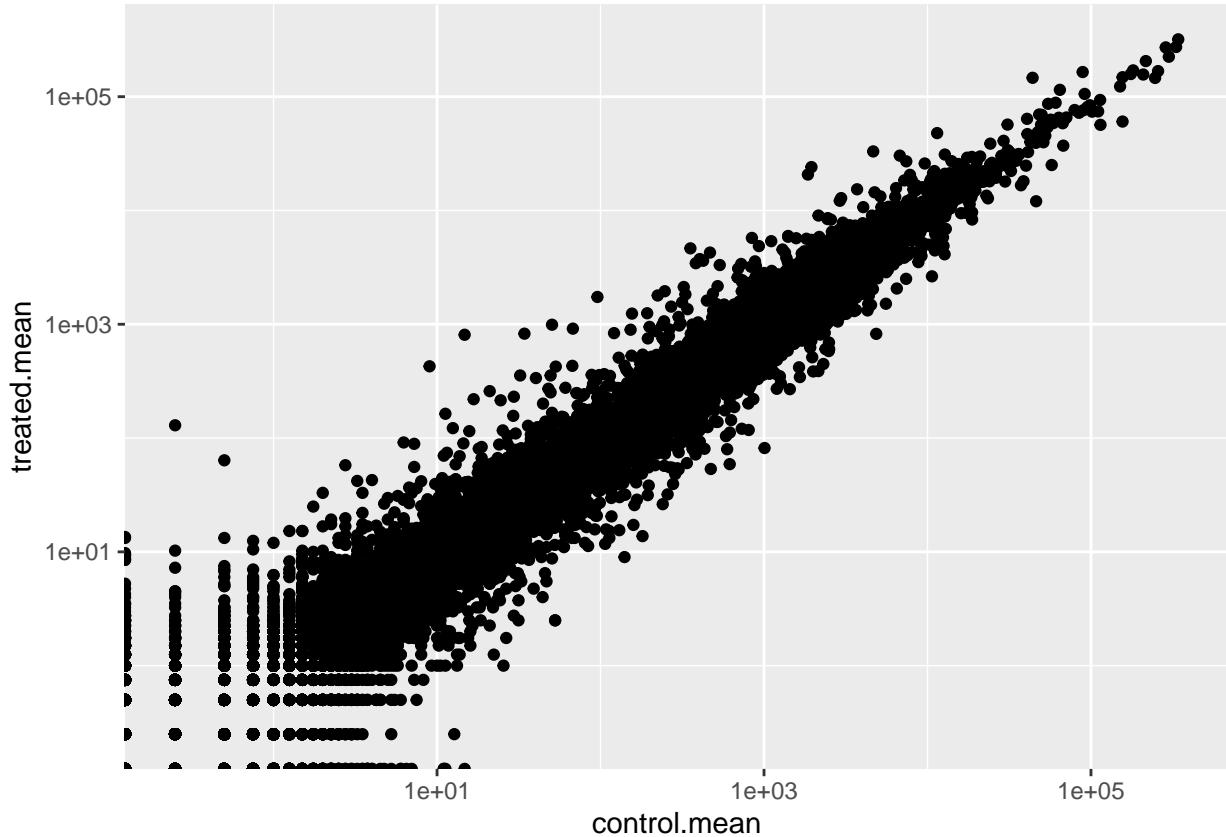
```

library(ggplot2)
ggplot(meancounts, aes(x=control.mean, y=treated.mean)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10()

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

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

```



We often use `log2` in this field because it have nice math properties that make interpretation easier.

```
log2(10)
```

```
## [1] 3.321928
```

```
log2(5/10)
```

```
## [1] -1
```

```
log2(40/10)
```

```
## [1] 2
```

Cool we see 0 values for no change and + values for increases and minus values for decreases. This nice property leads us to work with `log2(fold-change)` all the time in the genomics and proteomics filed.

Let's add the `log2(fold-change)` values to our 'meancounts' dataframe

```
meancounts$log2fc <- log2(meancounts[, "treated.mean"]/
                                meancounts[, "control.mean"])
head(meancounts)
```

```

##                               control.mean treated.mean      log2fc
## ENSG000000000003      900.75      658.00 -0.45303916
## ENSG000000000005      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

```

I need to exclude the genes (i.e. rows) with zero counts because we can't say anything about these as we have no data for them!

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?

```
head(meancounts[, 1:2])
```

```

##                               control.mean treated.mean
## ENSG000000000003      900.75      658.00
## ENSG000000000005      0.00       0.00
## ENSG00000000419     520.50      546.00
## ENSG00000000457     339.75      316.50
## ENSG00000000460      97.25       78.75
## ENSG00000000938      0.75       0.00

```

```
head(meancounts[, 1:2]==0)
```

```

##                               control.mean treated.mean
## ENSG000000000003      FALSE      FALSE
## ENSG000000000005      TRUE       TRUE
## ENSG00000000419      FALSE      FALSE
## ENSG00000000457      FALSE      FALSE
## ENSG00000000460      FALSE      FALSE
## ENSG00000000938      FALSE      TRUE

```

```
which( c(F,F,T,T))
```

```
## [1] 3 4
```

I can use the **which()** function with the 'are.ind=TRUE' argument to get the columns and rows where the TRUE values are (i.e. the zero counts in our case).

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

```

##                               row col
## ENSG000000000005      2   1
## ENSG00000004848     65   1
## ENSG00000004948     70   1
## ENSG00000005001     73   1
## ENSG00000006059    121   1
## ENSG00000006071    123   1

```

```
to.rm <- unique(zero.vals[, "row"])
head(sort(to.rm))
```

```
## [1] 2 6 65 70 73 81
```

Now remove these from our ‘meancounts’ dataframe.

```
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
```

How many do we have left

```
nrow(mycounts)
```

```
## [1] 21817
```

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

```
up.ind <- mycounts$log2fc > 2
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?

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

```
## [1] 367
```

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

We should not trust these results because the data was analyzed with a fold change. In order to trust these results, we should determine whether differences in data are statistically significant.

```
#DESeq2 analysis
```

```

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, 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

## 
## Attaching package: 'IRanges'

## The following object is masked from 'package:grDevices':
## 
##     windows

## 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, colAnyNs, 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, rowAnyNs, 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

citation("DESeq2")

## 
## 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},
## }

```

We need to first setup the input for DESeq

```

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

```

Now we can run DESeq analysis

results() first produces an error, need to run DESeq first

```

dds <- DESeq(dds)

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

```

To get at the results here we use the DESeq ‘results()’ function:

```

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

```

```
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

```

```
##Save our results
```

Write out whole results dataset (including genes that do not change significantly)

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

Focus in on those genes with a small p-value (i.e. show a significant change).

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

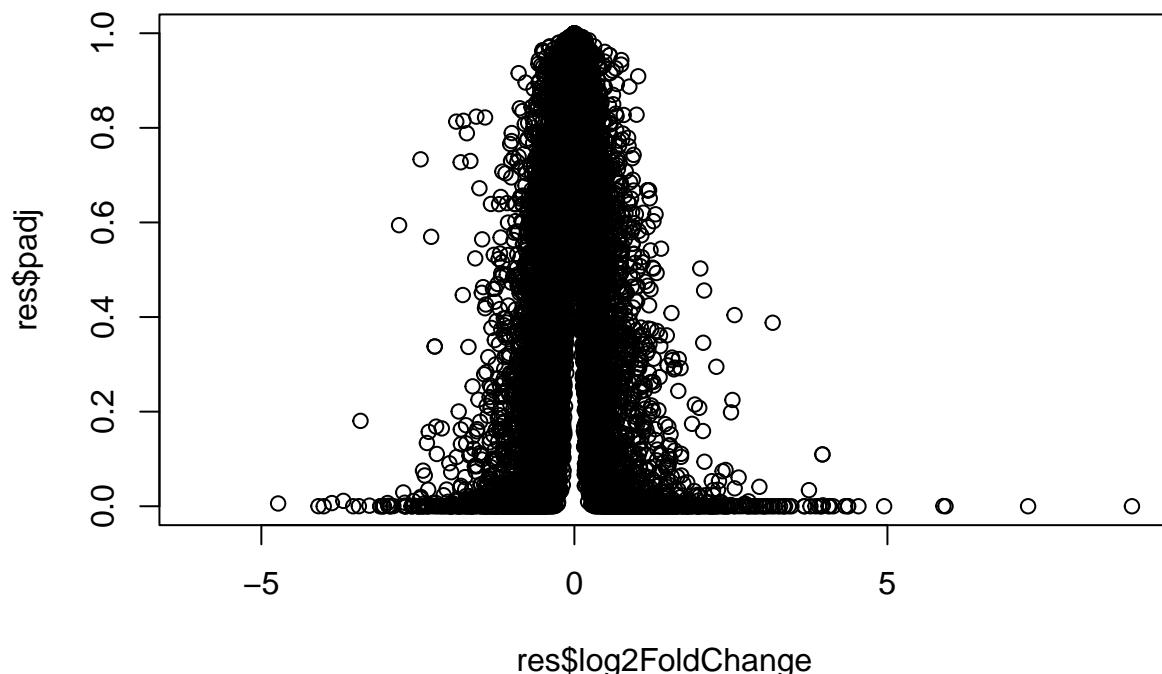
```
summary(res05)
```

```
##  
## 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
```

```
#data visualization - volcano plot
```

Let's make a commonly produced visualization from this data, namely a so-called Volcano plot. These summary figures are frequently used to highlight the proportion of genes that are both significantly regulated and display a high fold change.

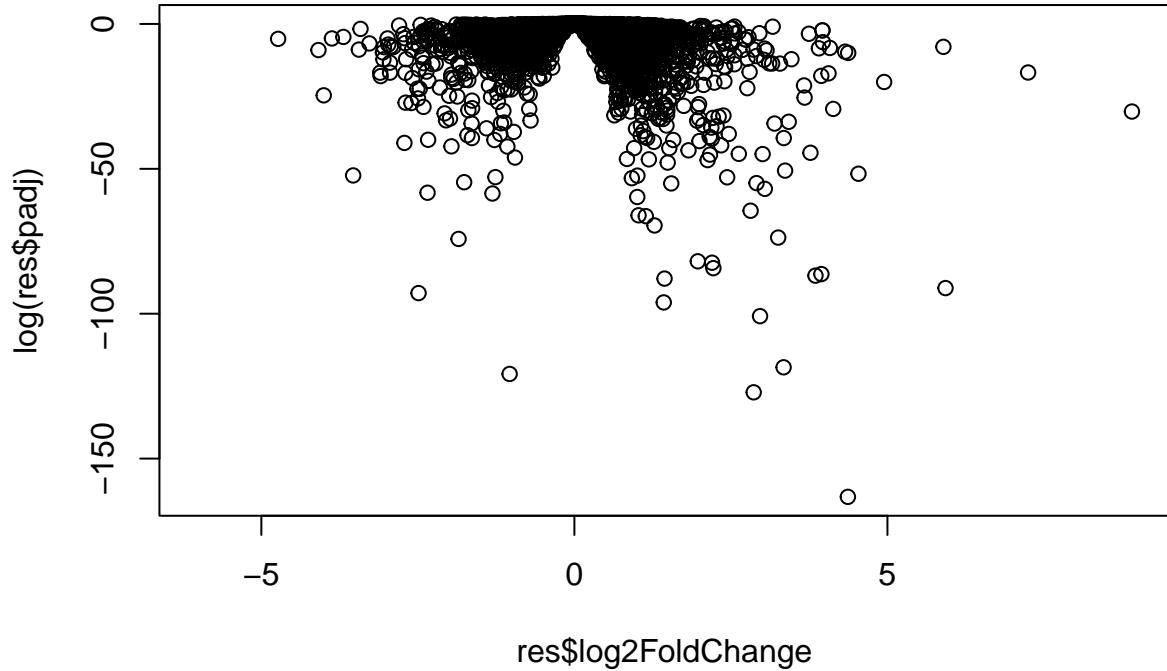
```
plot(res$log2FoldChange, res$padj)
```



That is not a useful plot because all the small p-values are hidden at the bottom of the plot and we can't really see them.

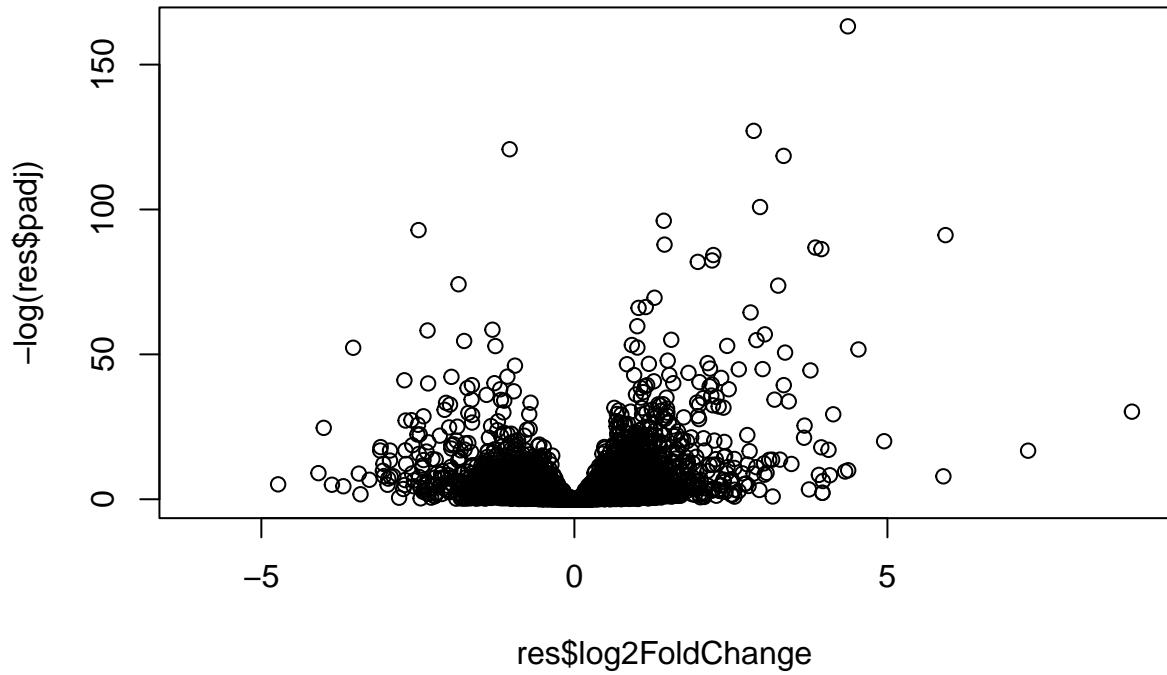
Log will help.

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



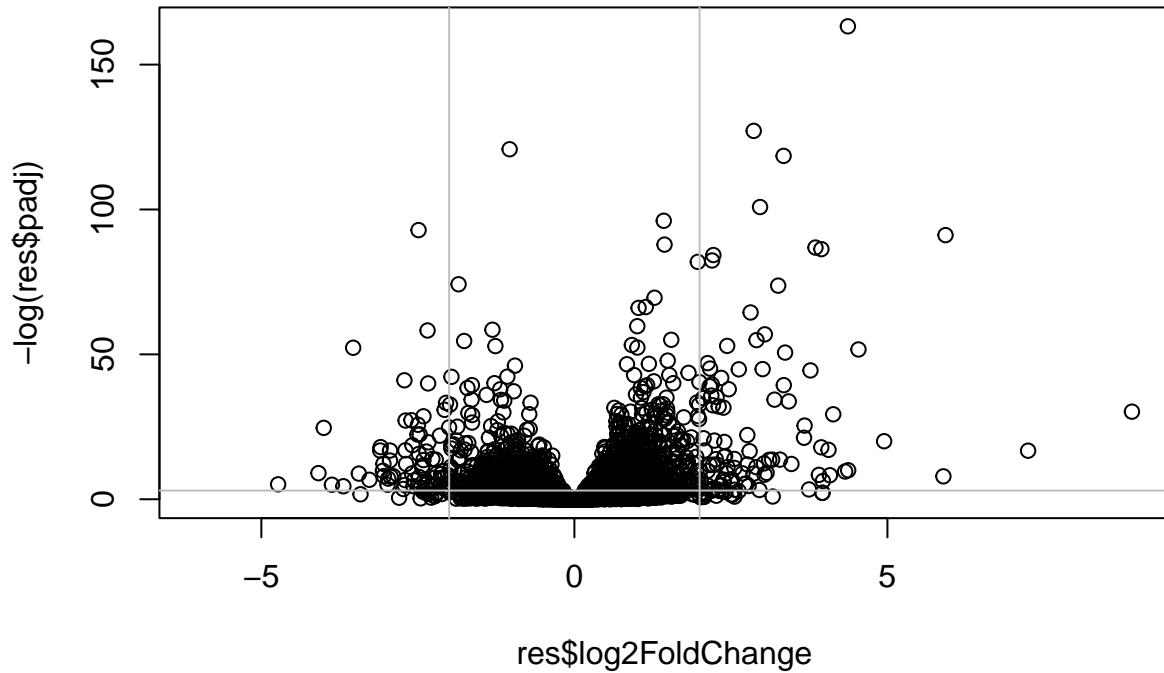
We can flip this p-value axis by just putting a minus sign on it then we will have the classic volcano plot that the rest of the world uses.

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



Finally let's add some color to this plot to draw attention to the genes (i.e. points) we care about - that is those with large fold-change and low p-values (i.e. high `-log(p-values)`)

```
plot(res$log2FoldChange, -log(res$padj))
abline(v=c(-2, +2), col = "gray")
abline(h=-log(0.05), col = "gray")
```



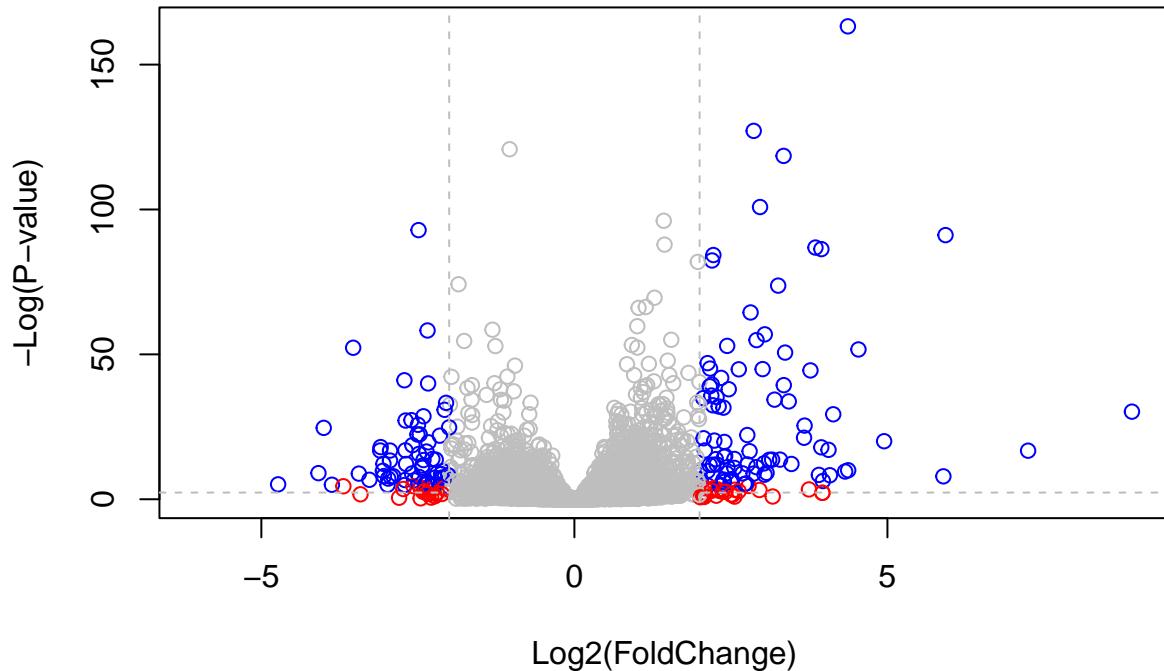
Now we can add some color to the points

```
#set up 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
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)
```



```
#Adding Annotation data for our genes
```

```
For this we need two bioconductor packages -BiocManager::install("AnnotationDbi") -BiocManager::install("org.Hs.eg.db")
```

```
#Using mapIDs() function from bioconductor
```

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

```
##
```

Let's have a look at what is in the 'org.Hs.eg.db'

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

```
## [1] "ACNUM"          "ALIAS"           "ENSEMBL"         "ENSEMLPROT"      "ENSEMLTRANS"
## [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 will use the 'mapIDs' function to translate between identifiers from different databases.

```

res$symbol <- mapIds(org.Hs.eg.db,
                      keys=row.names(res), # Our genenames
                      keytype="ENSEMBL", # The format
                      column="SYMBOL", # new format we want
                      multiVals="first")

```

```

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

```

We need ENTREZ ids or pathway analysis with KEGG.

```

columns(org.Hs.eg.db)

```

```

## [1] "ACNUM"          "ALIAS"           "ENSEMBL"         "ENSEMLPROT"      "ENSEMLTRANS"
## [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"

```

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

```

res$entrez <- mapIds(org.Hs.eg.db,
                      keys=row.names(res), # Our genenames
                      keytype="ENSEMBL", # The format
                      column="ENTREZID", # new format we want
                      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 8 columns
##           baseMean log2FoldChange      lfcSE      stat     pvalue
##           <numeric>      <numeric> <numeric> <numeric> <numeric>
## ENSG000000000003  747.194195   -0.3507030  0.168246 -2.084470 0.0371175
## ENSG000000000005   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
##           <numeric> <character> <character>
## ENSG000000000003   0.163035     TSPAN6      7105
## ENSG000000000005      NA        TNMD      64102
## ENSG00000000419   0.176032     DPM1       8813
## ENSG00000000457   0.961694     SCYL3      57147
## ENSG00000000460   0.815849    C1orf112     55732
## ENSG00000000938      NA        FGR       2268

```

Let's make another volcano plot with some gene labels. For this we can use the EnhancedVolcano package

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

## Bioconductor version 3.14 (BiocManager 1.30.16), R 4.1.1 (2021-08-10)

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

## Installation paths not writeable, unable to update packages
##   path: C:/Program Files/R/R-4.1.1/library
##   packages:
##     lattice, mgcv, nlme, survival

## Old packages: 'BiocParallel', 'cli', 'corrplot', 'cpp11', 'crayon',
##   'crosstalk', 'generics', 'GenomicRanges', 'gert', 'glue', 'pillar',
##   'RcppArmadillo', 'readr', 'rlang', 'tibble', 'tinytex', 'tzdb', 'usethis',
##   'vroom', 'xfun'

library(EnhancedVolcano)

## Loading required package: ggrepel

## Registered S3 methods overwritten by 'ggalt':
##   method           from
##   grid.draw.absoluteGrob  ggplot2
##   grobHeight.absoluteGrob ggplot2
##   grobWidth.absoluteGrob ggplot2
##   grobX.absoluteGrob    ggplot2
##   grobY.absoluteGrob    ggplot2

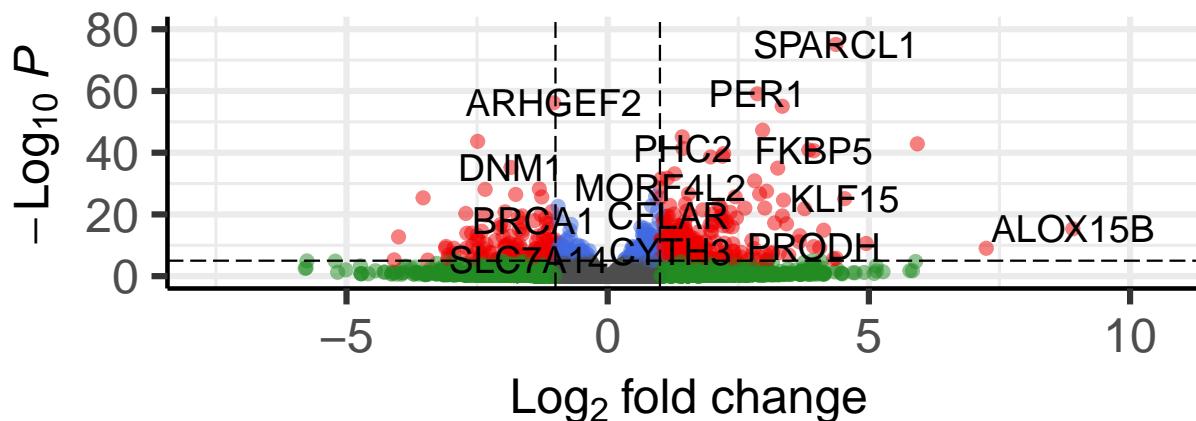
x <- as.data.frame(res)

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

# Volcano plot

*EnhancedVolcano*

● NS ● Log<sub>2</sub> FC ● p-value ● p-value and log<sub>2</sub> FC



total = 38694 variables

```
#Pathway analysis/geneset annotation
```

```
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)
```

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

```
#look at first two pathways
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"
```

The main `gage()` function requires a named vector of fold changes, where the names of the values are the Ensembl gene IDs

```
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

# Get the results
keggres = gage(foldchanges, gsets=kegg.sets.hs)
```

```
attributes(keggres)
```

```
## $names
## [1] "greater" "less"    "stats"
```

This separates out results by “greater” and “less” i.e., those that are up regulated and those that are down regulated.

```
# Look at the first three down (less) pathways
head(keggres$less, 3)
```

```
##                                     p.geomean stat.mean      p.val
## hsa05332 Graft-versus-host disease 0.0004250461 -3.473346 0.0004250461
## hsa04940 Type I diabetes mellitus 0.0017820293 -3.002352 0.0017820293
## hsa05310 Asthma                  0.0020045888 -3.009050 0.0020045888
##                                     q.val set.size      exp1
## hsa05332 Graft-versus-host disease 0.09053483      40 0.0004250461
## hsa04940 Type I diabetes mellitus 0.14232581      42 0.0017820293
## hsa05310 Asthma                  0.14232581      29 0.0020045888
```

Now, let’s try out the `pathview()` function from the `pathview` package to make a pathway plot with our RNA-Seq expression results shown in color.

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

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

```
## Info: Working in directory C:/Users/Nicole/Documents/bimm143_github/class15
```

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

