

Class 13: RNA Seq (pt.1)

Keilyn Duarte (PID A16881868)

Installed packages BiocManager and DESeq2

```
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
```

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

Import countData and colData

There are two datasets I need to import/read

- `countData` the transcript counts per gene (rows) in the different experiments
- `colData` information (aka metadata) about the columns (i.e. experiments) in `countData`.

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

Let's look at some of these datasets with `head()`

```
head(counts)
```

	SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516
ENSG000000000003	723	486	904	445	1170
ENSG000000000005	0	0	0	0	0
ENSG000000000419	467	523	616	371	582
ENSG000000000457	347	258	364	237	318
ENSG000000000460	96	81	73	66	118
ENSG000000000938	0	0	1	0	2
	SRR1039517	SRR1039520	SRR1039521		
ENSG000000000003	1097	806	604		
ENSG000000000005	0	0	0		
ENSG000000000419	781	417	509		
ENSG000000000457	447	330	324		
ENSG000000000460	94	102	74		
ENSG000000000938	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

Q2. How many ‘control’ cell lines do we have?

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

[1] 4

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

- Extract all “control” columns

```

control inds <- metadata$dex == "control"
control counts <- counts[, control inds]
head(control counts)

```

	SRR1039508	SRR1039512	SRR1039516	SRR1039520
ENSG000000000003	723	904	1170	806
ENSG000000000005	0	0	0	0
ENSG000000000419	467	616	582	417
ENSG000000000457	347	364	318	330
ENSG000000000460	96	73	118	102
ENSG000000000938	0	1	2	0

- Find the row wise mean value for each gene in these columns

```

control mean <- rowSums(control counts)/4
head(control mean)

```

ENSG000000000003	ENSG000000000005	ENSG000000000419	ENSG000000000457	ENSG000000000460
900.75	0.00	520.50	339.75	97.25
ENSG000000000938				
0.75				

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

You can make the number of columns a vector, so that it can be changed depending on the number of experiments

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)

- Extract all “treated” columns

```

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

```

	SRR1039509	SRR1039513	SRR1039517	SRR1039521
ENSG000000000003	486	445	1097	604
ENSG000000000005	0	0	0	0
ENSG000000000419	523	371	781	509
ENSG000000000457	258	237	447	324
ENSG000000000460	81	66	94	74
ENSG000000000938	0	0	0	0

- Find the row wise mean value for each gene in these columns

```
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
```

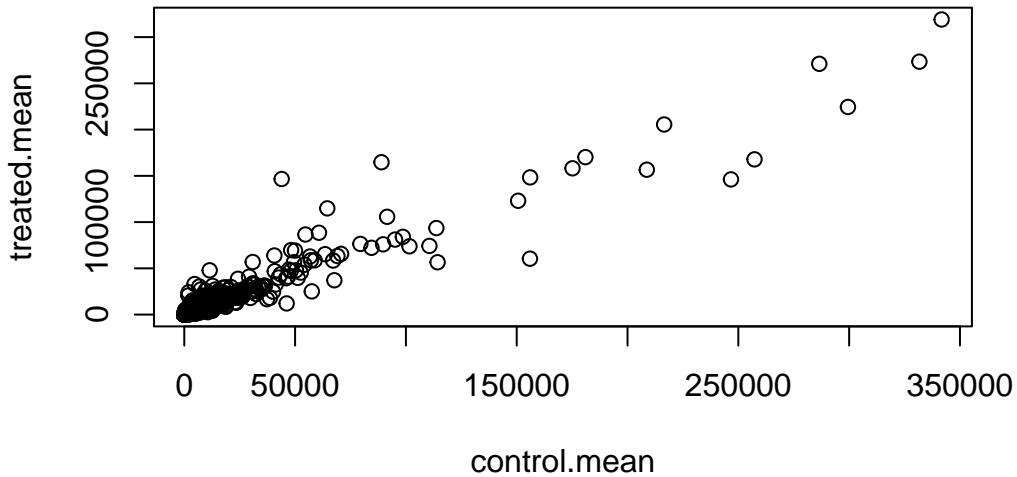
Let's put these two mean values together for easy book-keeping

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

Q5 (a). Create a scatter plot showing the mean of the treated samples against the mean of the control samples.

Let's have a look - i.e. plot control.mean vs. treated.mean

```
plot(meancounts)
```



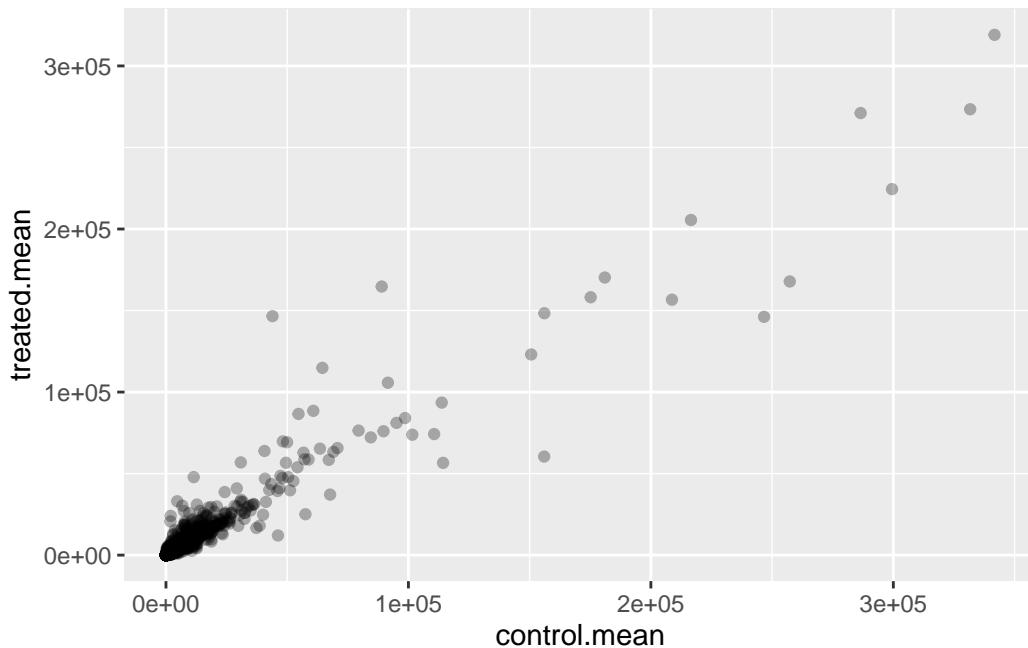
(b). You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot?

```

library(ggplot2)

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

```



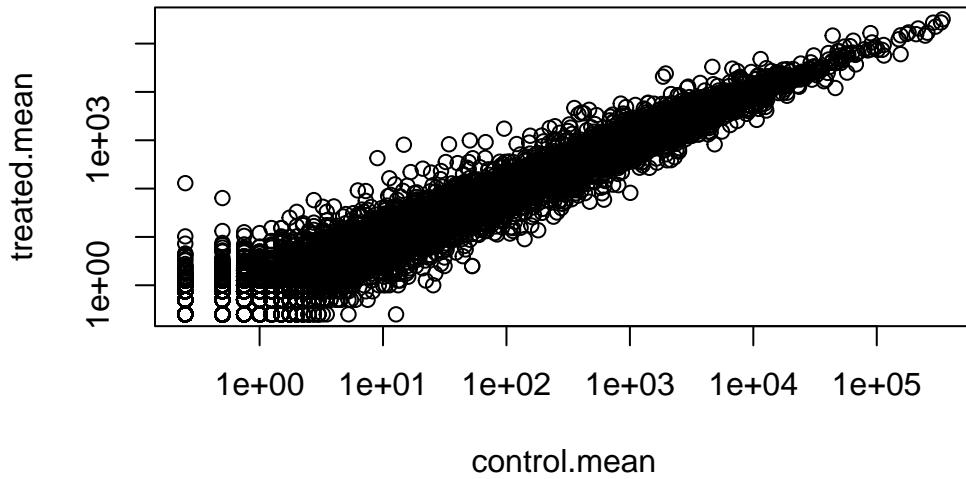
Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

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

```
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



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

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

```
[1] 0
```

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

```
[1] 1
```

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

```
[1] 2
```

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

```
[1] -1
```

A positive value indicates that there was an increase in gene expression once treated, and a negative value indicates that there was a decrease in gene expression once treated.

We can now add “log2 fold-change” values to our `meancounts` dataset.

```
# Adding a column `log2fc` to the `meancounts` dataset (using the $ sign)
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
ENSG000000000419	520.50	546.00	0.06900279
ENSG000000000457	339.75	316.50	-0.10226805
ENSG000000000460	97.25	78.75	-0.30441833
ENSG000000000938	0.75	0.00	-Inf

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

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?

```
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
ENSG000000000003	900.75	658.00	-0.45303916
ENSG000000000419	520.50	546.00	0.06900279
ENSG000000000457	339.75	316.50	-0.10226805
ENSG000000000460	97.25	78.75	-0.30441833
ENSG000000000971	5219.00	6687.50	0.35769358
ENSG000000001036	2327.00	1785.75	-0.38194109

The arr.ind argument recalls only rows and columns from the `meancounts` that are TRUE for having zero values. The unique function makes sure that there's no repeats if there are multiple TRUE / zeros in any of the columns for that row.

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

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
```

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

How many genes are “down” regulated at the threshold of -2

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

```
[1] 2330
```

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

Not yet, because we have to evaluate the statistical significance first.

DESeq2

To do this the right way we need to consider the 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 <- 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

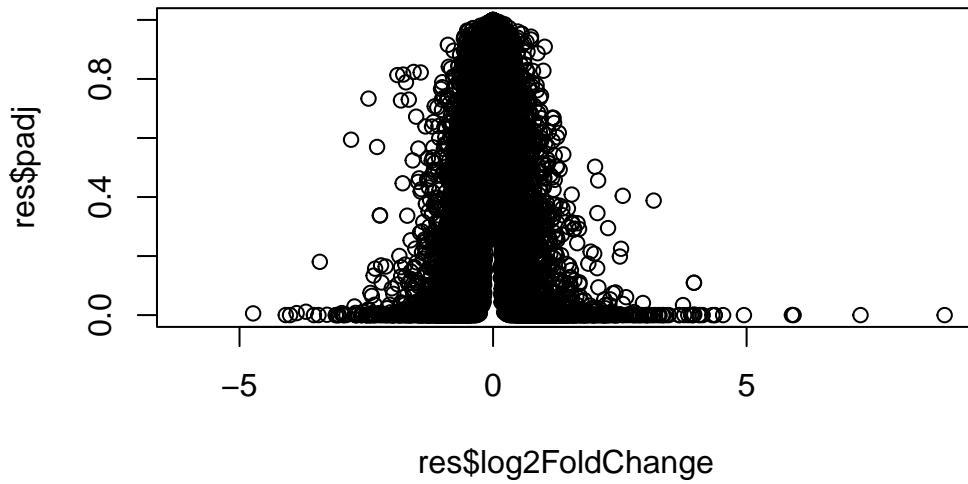
DataFrame with 6 rows and 6 columns

	baseMean	log2FoldChange	lfcSE	stat	pvalue
	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>
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
	<numeric>
ENSG000000000003	0.163035
ENSG000000000005	NA
ENSG00000000419	0.176032
ENSG00000000457	0.961694
ENSG00000000460	0.815849
ENSG00000000938	NA

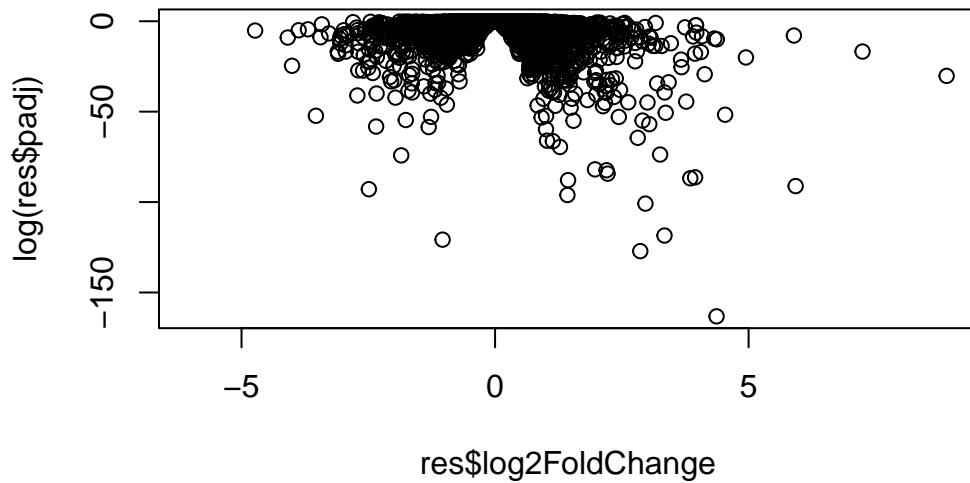
Plot of fold-change vs P-value (adjusted for multiple testing)

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



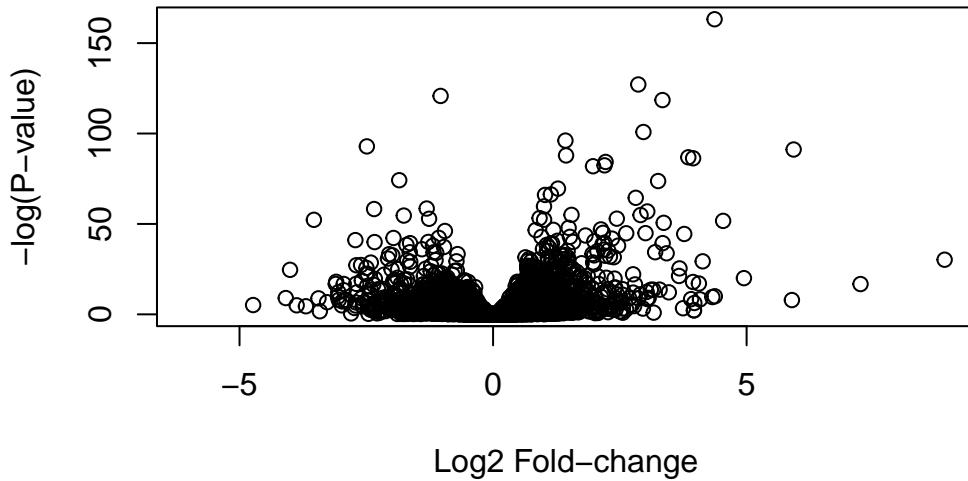
Take the log of the P-value

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



We can just flip the axis to see better

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



Let's save our work to date

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

To finish off, let's make a nicer volcano plot.

- Add the log2 threshold lines of +2 and -2
- Add p-value threshold lines at 0.05
- Add color to highlight the subset of genes that meet both of the above thresholds.

Make mycols vector for the plot

```
# Replicate gray with `rep()` by the number of rows in `res`
mycols <- rep("gray", nrow(res))
# If log2FoldChange in res is more than or equal to 2, make the color red
mycols[ abs(res$log2FoldChange) >= 2 ] <- "red"
# If log2FoldChange in res is less than or equal to -2, make the color blue
mycols[res$log2FoldChange <= -2] <- "blue"
# If the padjusted value is higher than 0.05, make the color black
mycols[res$padj > 0.05] <- "black"
```

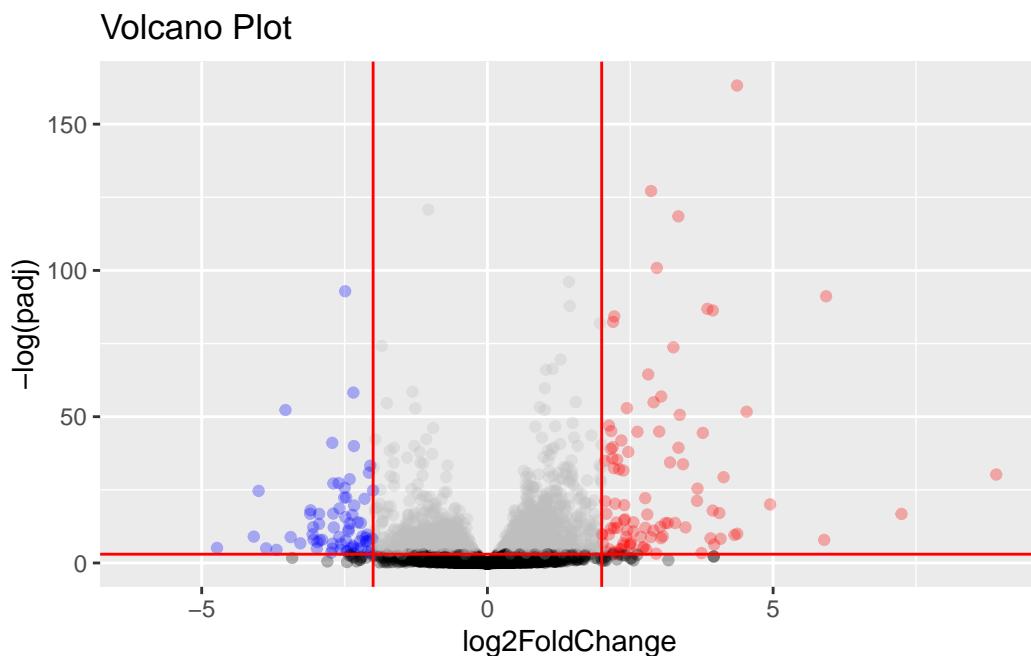
Make it with ggplot:

```

ggplot(res) +
  aes(log2FoldChange, -log(padj)) +
  geom_point(alpha = 0.3, col=mycols) +
  geom_vline(xintercept=c(-2,2), col="red") +
  geom_hline(yintercept= -log(0.05), col="red") +
  labs(title="Volcano Plot")

```

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



Add gene annotation data

Now the question is what are the blue and red points in the above volcano plot - i.e. what are the genes most influenced by drug treatment?

We will use some BioConductor packages to “map” the ENSEMBLE ids to more useful gene SYMBOL names/ids.

We can install these packages with `BiocManager::install("AnnotationDbi")`

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

What database identifiers can I translate between here:

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

```
[1] "ACCCNUM"      "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 now use the `mapIds()` function to translate/map between these different identifier formats

```
res$symbol <- mapIds(org.Hs.eg.db,
                      keys = rownames(res),
                      keytype = "ENSEMBL",
                      column = "SYMBOL")
```

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

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

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

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

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

Pathway Analysis

Now I know the gene names and their IDs in different databases. I want to know what type of biology they are involved in...

This is the job of “pathway analysis” (aka “gene set enrichment”)

There are tons of different BioConductor packages for pathway analysis. Here we use just one of them called `gage` and `pathview`. I will install these packages with `BiocManager::install(c("gage", "pathview", "gageData"))`

```
library(gage)
```

```
library(gageData)
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>).

```
#####
```

Load up the datasets

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

```
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"
```

We will use these KEGG genesets (aka pathways) and our `res` results to see what overlaps. To do this we will use the `gage()` function.

For input `gage()` wants just a vector of importance - in our case Foldchange values.

```
foldchanges <- res$log2FoldChange
```

Vectors in R can have “names” that are useful for bookkeeping, so we know what a given value corresponds to.

Lets put names on our `foldchanges` vector - here we will use `res$entrez`

```
names(foldchanges) <- res$entrez
```

Now we can run “pathway analysis”

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

head(keggres$less)
```

	p.geomean	stat.mean
hsa05332 Graft-versus-host disease	0.0004250461	-3.473346
hsa04940 Type I diabetes mellitus	0.0017820293	-3.002352
hsa05310 Asthma	0.0020045888	-3.009050
hsa04672 Intestinal immune network for IgA production	0.0060434515	-2.560547
hsa05330 Allograft rejection	0.0073678825	-2.501419
hsa04340 Hedgehog signaling pathway	0.0133239547	-2.248547
	p.val	q.val
hsa05332 Graft-versus-host disease	0.0004250461	0.09053483
hsa04940 Type I diabetes mellitus	0.0017820293	0.14232581
hsa05310 Asthma	0.0020045888	0.14232581
hsa04672 Intestinal immune network for IgA production	0.0060434515	0.31387180
hsa05330 Allograft rejection	0.0073678825	0.31387180
hsa04340 Hedgehog signaling pathway	0.0133239547	0.47300039

	set.size	exp1
hsa05332 Graft-versus-host disease	40	0.0004250461
hsa04940 Type I diabetes mellitus	42	0.0017820293
hsa05310 Asthma	29	0.0020045888
hsa04672 Intestinal immune network for IgA production	47	0.0060434515
hsa05330 Allograft rejection	36	0.0073678825
hsa04340 Hedgehog signaling pathway	56	0.0133239547

We can get a pathway image file with our genesets highlighted via the `pathview()` function.

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

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

```
Info: Working in directory /Users/keilynduarte/Desktop/BIMM 143/Class13
```

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

Insert this figure in my report

