

class 13 RNA-Seq with DESeq2

Melandra Aboueid (PID: 17473102)

Table of contents

Background	1
Data Import	1
differential gene expression	3
DESeq analysis	8
Principal Component Analysis (PCA)	11
Run the DESeq analysis pipeline	13
Volcano Plot	14
Setup our custom point color vector	16
Save our results	18
Adding annotation data	18
Save annotated results to a CSV file	21
Volcano plot	22
Pathway analysis	23

Background

Today we will perform an RNAseq analysis of the effects of a common steroid on airway cells.

In particular, dexamethasone (hereafter just called “dex”) on different airway smooth muscle cell lines (ASM cells).

Data Import

We need two different inputs:

- **countData:** With genes in rows and experiments in columns
- **colData:** meta data that describes the columns in countData

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

```
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		

```
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

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
```

```
table(metadata$dex)
```

```
control treated
      4       4
```

differential gene expression

We have 4 replicate drug treated and control (no drug) columns/experiments in our `counts` object.

We want one “mean” value for each gene (rows) in “treated” (drug) and one mean value for each gene in “control” cols.

Step 1: Find all “control” columns in `counts`

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

Step 2: Extract these columns to a new object called `control.counts`

```
control counts <- counts[ , control inds]
```

Step 3: then calculate the mean value for each gene

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

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

We could make the code more robust by using the `rowMeans()` function instead of manually summing and dividing so we don't hard-code /4.

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`)

Now do the same thing for the “treated” columns/experiments...

Step 1: Find all “treated” columns in `counts`

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

Step 2: Extract these columns to a new object called `treated.counts`

```
treated counts <- counts[ , treated inds]
```

Step 3: then calculate the mean value for each gene

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

put these together for easy bookkeeping as `meancounts`

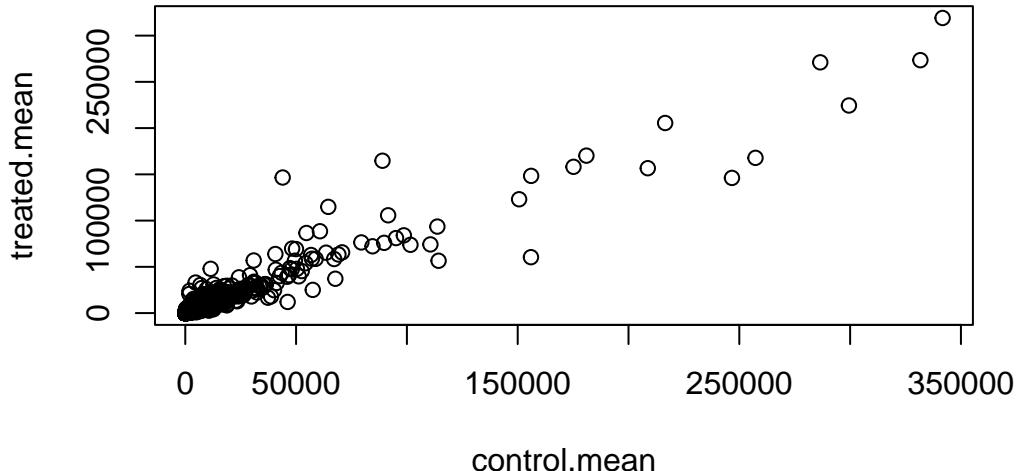
```
meancounts <- data.frame(control mean, treated mean)
```

Q5 (a). Create a scatter plot showing the mean of the treated samples against the mean of the control samples. Your plot should look something like the following.

explicitly doing `plot(control.mean, treated.mean)` makes it crystal-clear you plotted “treated vs control,” but `plot(meancounts)` is still fine.

A quick plot

```
plot(meancounts)
```

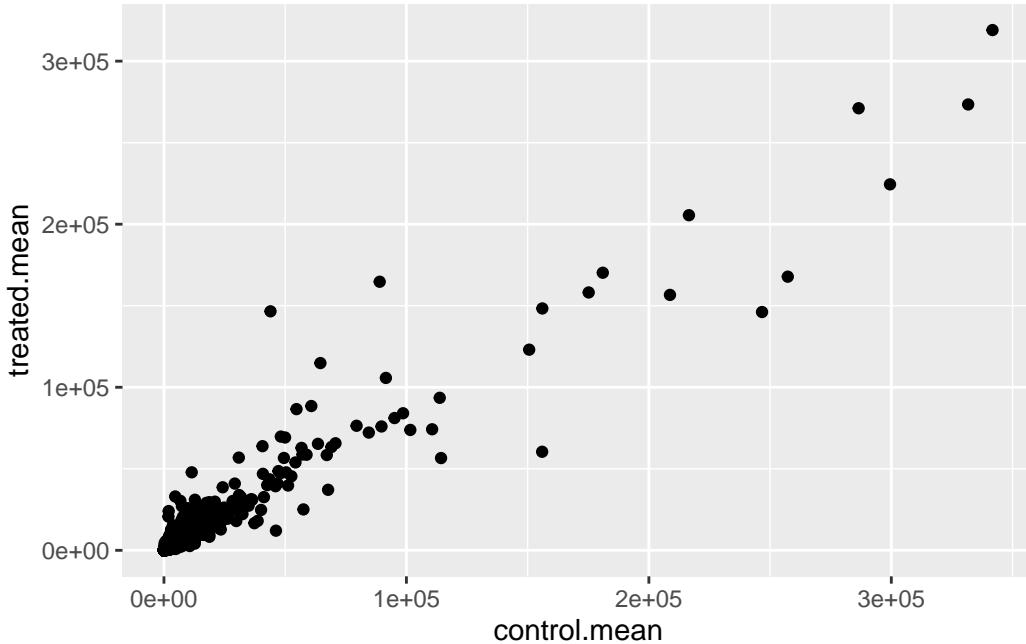


Q5 (b). You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot?

The geom used for this scatter plot is geom_point().

```
library(ggplot2)

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



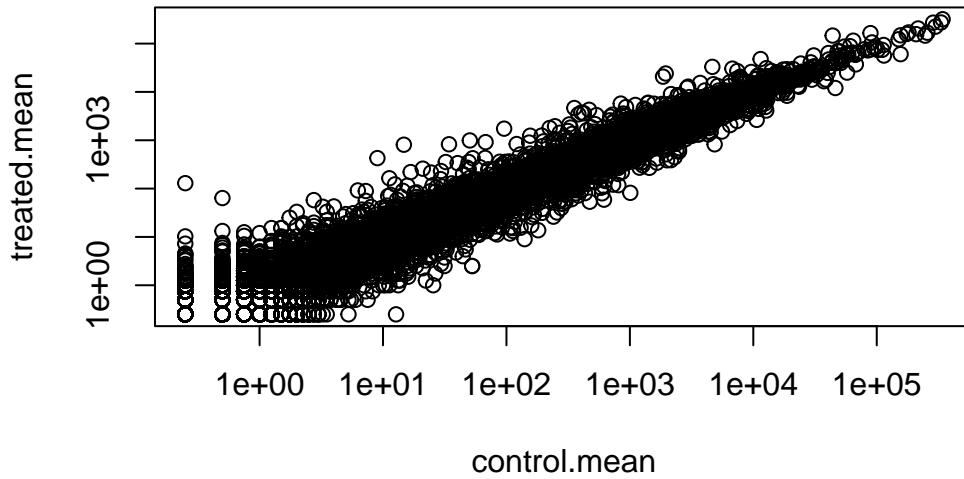
Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

Lets log trasfrom this count data

```
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



N.B. We most often use log2 for this type of data so it makes the interpretation much more straightforward.

Treated/Control is often called “fold-change”

if there was no change we would have a log2-fc of zero:

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

```
[1] 0
```

If we had double the amount of transcript around we would have a log2-fc of 1

```
log2(20/10)
```

```
[1] 1
```

If we had half as much transcript around we would have a log2-fc of -1

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

```
[1] -1
```

Q. Calculate a log2 fold change value for all of our genes and add it as a new column to our `meancounts` object.

```
meancounts$log2f <- log2(meancounts$treated.mean / meancounts$control.mean)

head(meancounts)
```

	control.mean	treated.mean	log2f
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

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

```
[1] 2
```

There are some “funky” log2fc values (NaN and -Inf) here that come about when ever we have 0 mean count values. Typically we would remove these genes from any further analysis - as we can’t say anything about them if 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?

`arr.ind=TRUE` returns row and column indices. `unique()` ensures each gene is removed only once.

```
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	log2f
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

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

```
up.ind <- mycounts$log2f > 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?

367

```
down.ind <- mycounts$log2f < -2  
sum(down.ind)
```

[1] 367

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

No, this method is not reliable because it ignores sequencing depth and variability. DESeq2 provides proper statistical testing.

DESeq analysis

lets do this analysis with an estimate of ststistical significance using the **DESeq2** package

```
library(DESeq2)
```

Loading required package: S4Vectors

Loading required package: stats4

Loading required package: BiocGenerics

Loading required package: generics

```
Attaching package: 'generics'
```

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

```
as.difftime, as.factor, as.ordered, intersect, is.element, setdiff,  
setequal, union
```

```
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, is.unsorted, lapply, Map, mapply, match, mget,  
order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,  
rbind, Reduce, rownames, sapply, saveRDS, table, tapply, 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: Seqinfo
```

```
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,
colDiffss, colIQRDiffss, colIQRs, colLogSumExps, colMadDiffss,
colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
colProds, colQuantiles, colRanges, colRanks, colSdDiffss, colSds,
colSums2, colTabulates, colVarDiffss, colVars, colWeightedMads,
colWeightedMeans, colWeightedMedians, colWeightedSds,
colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
rowCumsums, rowDiffss, rowIQRDiffss, rowIQRs, rowLogSumExps,
rowMadDiffss, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
rowSdDiffss, rowSds, rowSums2, rowTabulates, rowVarDiffss, 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
```

DESeq (like many bioconductor packages) want it's input data in a very specific way.

```
dds <- DESeqDataSetFromMatrix(countData = counts,
                                colData = metadata,
                                design = ~dex)
```

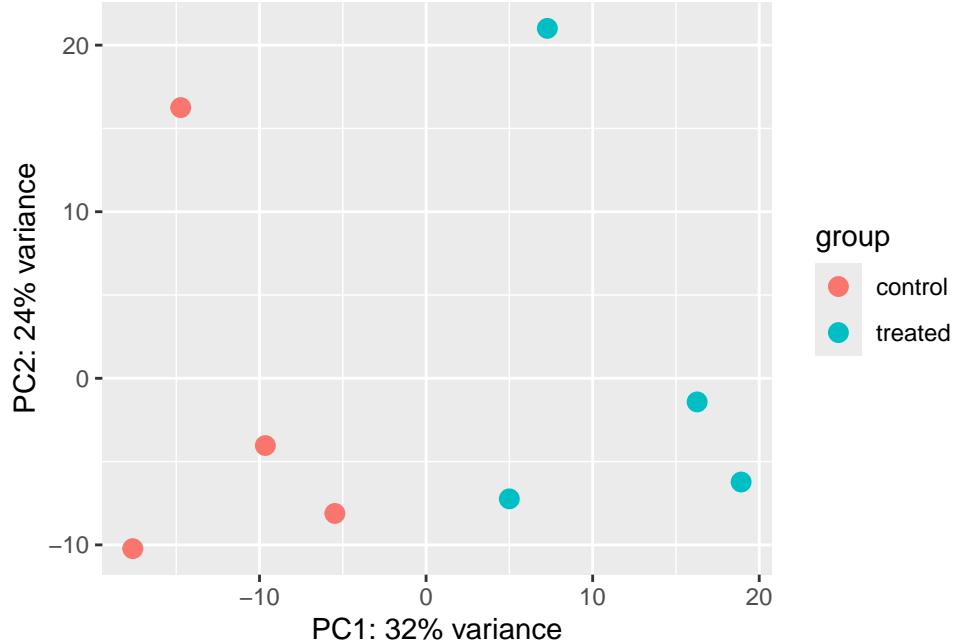
```
converting counts to integer mode
```

```
Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
design formula are characters, converting to factors
```

Principal Component Analysis (PCA)

```
vsd <- vst(dds, blind = FALSE)
plotPCA(vsd, intgroup = c("dex"))
```

```
using ntop=500 top features by variance
```



```
pcaData <- plotPCA(vsd, intgroup=c("dex"), returnData=TRUE)
```

using ntop=500 top features by variance

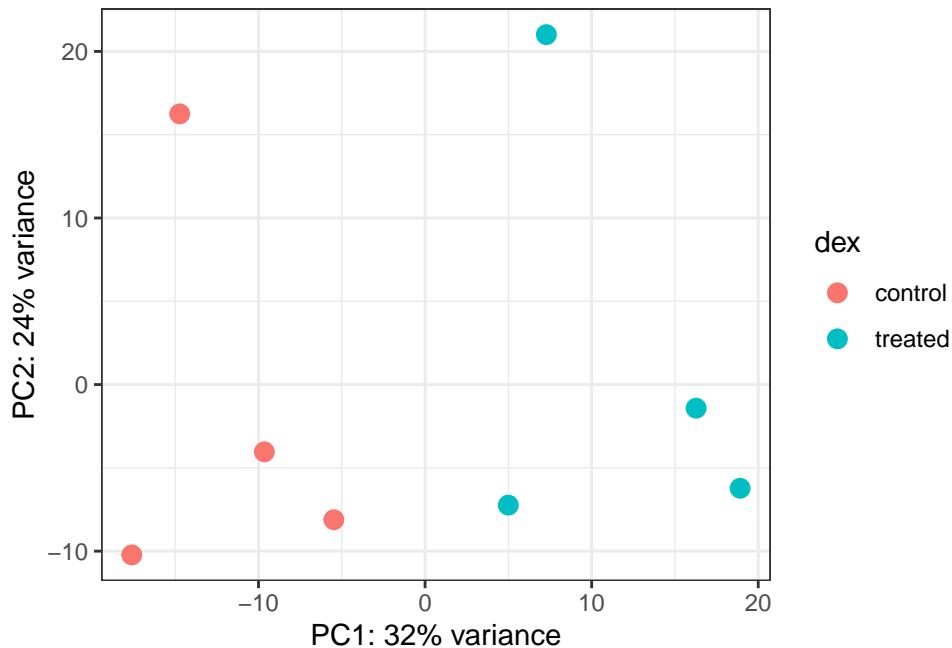
```
head(pcaData)
```

	PC1	PC2	group	name	id	dex	celltype
SRR1039508	-17.607922	-10.225252	control	SRR1039508	SRR1039508	control	N61311
SRR1039509	4.996738	-7.238117	treated	SRR1039509	SRR1039509	treated	N61311
SRR1039512	-5.474456	-8.113993	control	SRR1039512	SRR1039512	control	N052611
SRR1039513	18.912974	-6.226041	treated	SRR1039513	SRR1039513	treated	N052611
SRR1039516	-14.729173	16.252000	control	SRR1039516	SRR1039516	control	N080611
SRR1039517	7.279863	21.008034	treated	SRR1039517	SRR1039517	treated	N080611
	geo_id	sizeFactor					
SRR1039508	GSM1275862	1.0193796					
SRR1039509	GSM1275863	0.9005653					
SRR1039512	GSM1275866	1.1784239					
SRR1039513	GSM1275867	0.6709854					
SRR1039516	GSM1275870	1.1731984					
SRR1039517	GSM1275871	1.3929361					

Calculate percent variance per PC for the plot axis labels

```
percentVar <- round(100 * attr(pcaData, "percentVar"))
```

```
ggplot(pcaData) +
  aes(x = PC1, y = PC2, color = dex) +
  geom_point(size = 3) +
  xlab(paste0("PC1: ", percentVar[1], "% variance")) +
  ylab(paste0("PC2: ", percentVar[2], "% variance")) +
  coord_fixed() +
  theme_bw()
```



Run the DESeq analysis pipeline

The main function `DESeq`

```
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)
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 -0.350703  0.168242 -2.084514 0.0371134
ENSG00000000005  0.000000      NA       NA       NA       NA
ENSG00000000419 520.134160  0.206107  0.101042  2.039828 0.0413675
ENSG00000000457 322.664844  0.024527  0.145134  0.168996 0.8658000
ENSG00000000460 87.682625 -0.147143  0.256995 -0.572550 0.5669497
ENSG00000000938 0.319167 -1.732289  3.493601 -0.495846 0.6200029
  padj
  <numeric>
ENSG00000000003 0.163017
ENSG00000000005  NA
ENSG00000000419 0.175937
ENSG00000000457 0.961682
ENSG00000000460 0.815805
ENSG00000000938  NA

```

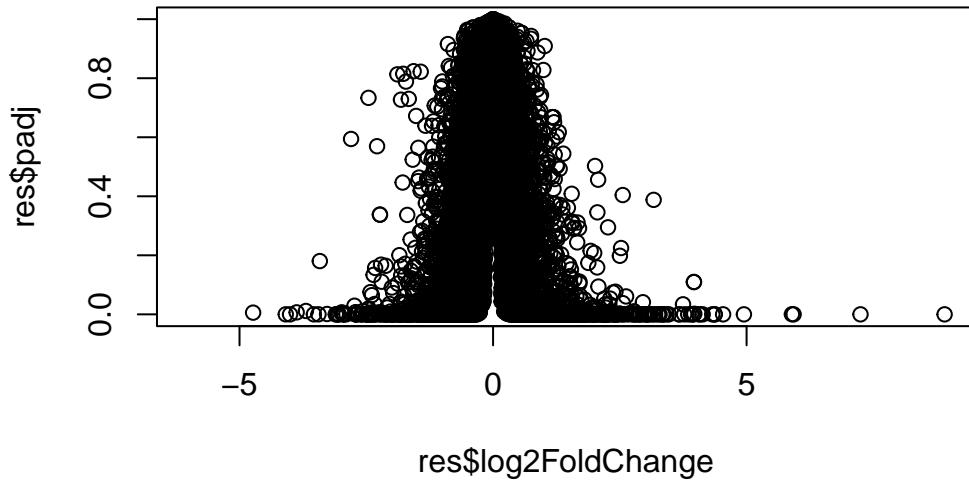
36000*0.05

[1] 1800

Volcano Plot

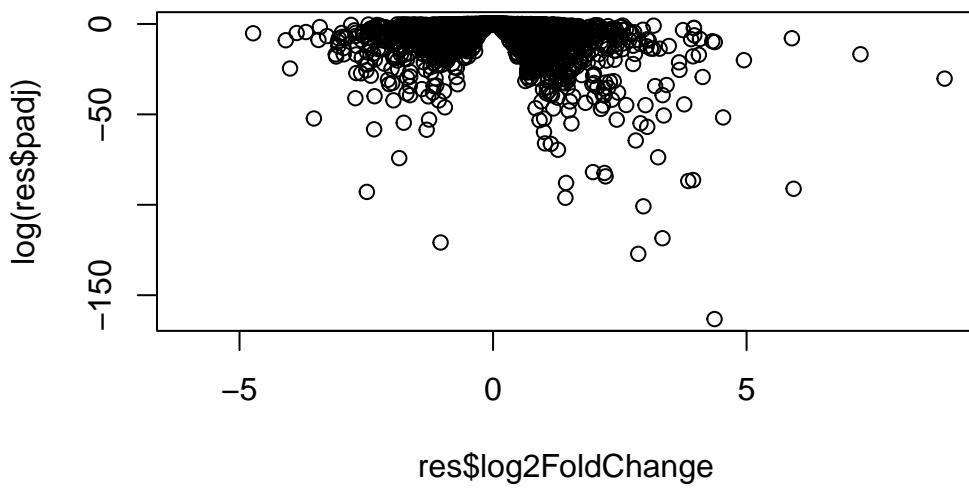
This is a main summary results figure from these kinds of studies. It is a plot of Log2-foldchange vs (Adjusted) P-Value

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



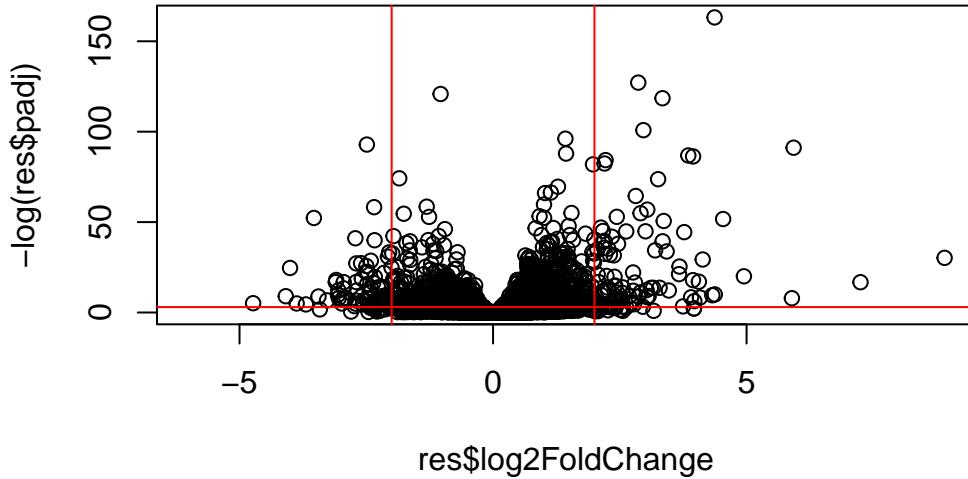
Again this y-axis is highly needs log transforming

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



Again this y-axis is highly needs log transforming and we can flip the y-axis with a minus sign so it looks like it

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



Setup our custom point color vector

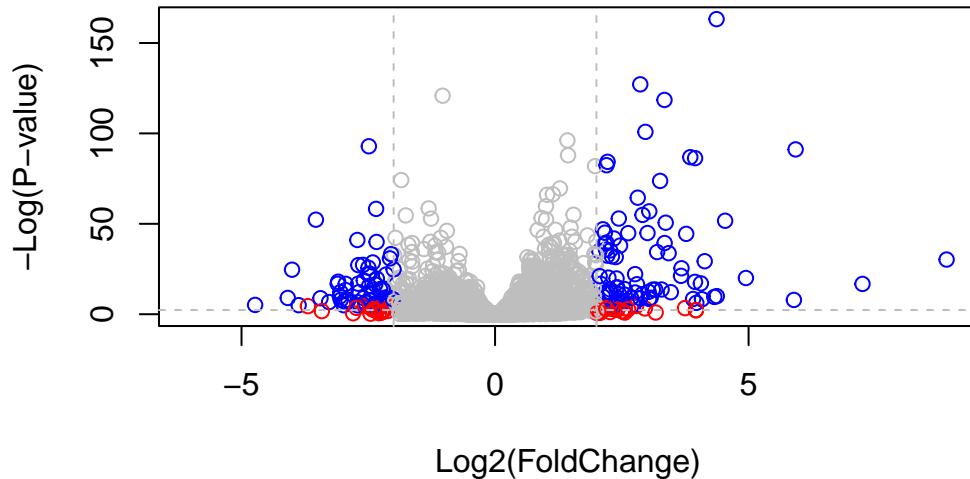
Start with default nase color “gray”

```
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 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)
```



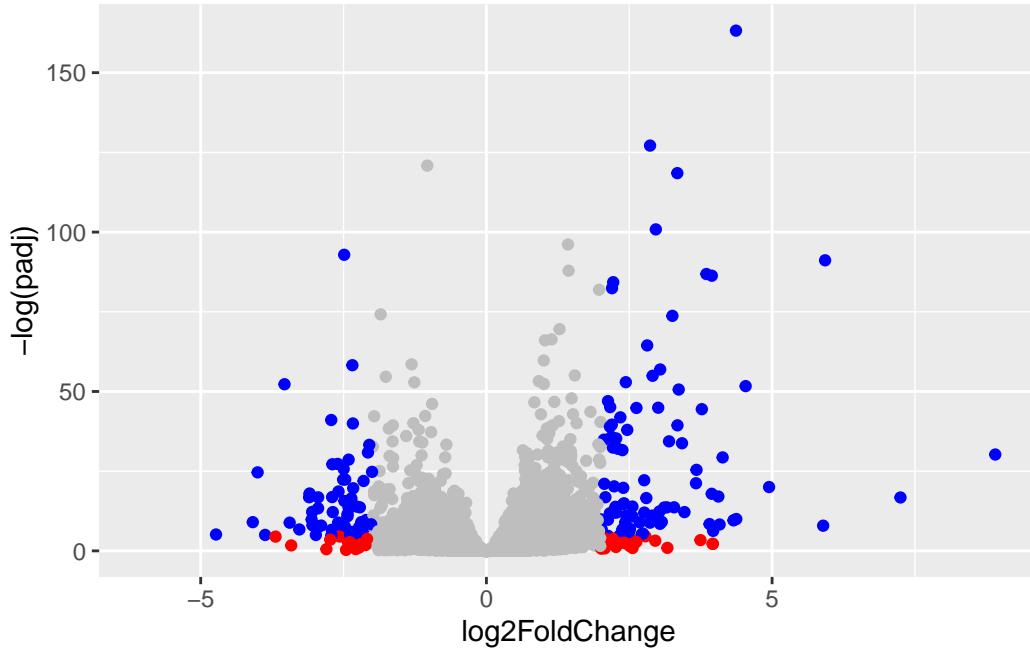
For even more customization you might find the EnhancedVolcano bioconductor package useful (Note. It uses ggplot under the hood):

Q. Make a presentation quality ggplot version of this plot. Include clear axis labels, a clean theme, your custom colors, cut-off line ans a plot title.

```
library(ggplot2)

ggplot(res) +
  aes(log2FoldChange,
      -log(padj)) +
  geom_point(colour = mycols)
```

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



```
labs(x="Log2 Fold-change",
     y="-log Adjusted p-value") +
  geom_vline(xintercept = c(-2,2)) +
  geom_hline(yintercept = -log(0.05)) +
  theme_bw()
```

NULL

Save our results

Adding annotation data

We need to add missing annotation data to our main res results object. this indicated common gene “symbol”

```
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
          0            0        0.0000000 0.0000000 0.0000000
          0            0        0.0000000 0.0000000 0.0000000
          0            0        0.0000000 0.0000000 0.0000000
          0            0        0.0000000 0.0000000 0.0000000
          0            0        0.0000000 0.0000000 0.0000000
          0            0        0.0000000 0.0000000 0.0000000
```

```

          <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG000000000003 747.194195 -0.350703  0.168242 -2.084514  0.0371134
ENSG000000000005 0.000000     NA        NA        NA        NA
ENSG000000000419 520.134160  0.206107  0.101042  2.039828  0.0413675
ENSG000000000457 322.664844  0.024527  0.145134  0.168996  0.8658000
ENSG000000000460 87.682625 -0.147143  0.256995 -0.572550  0.5669497
ENSG000000000938 0.319167 -1.732289  3.493601 -0.495846  0.6200029

          padj
          <numeric>
ENSG000000000003 0.163017
ENSG000000000005     NA
ENSG000000000419 0.175937
ENSG000000000457 0.961682
ENSG000000000460 0.815805
ENSG000000000938     NA

```

We will used R and bioconductor to do this “ID mapping”

```

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

```

Lets see what databases we can use for translocation/mapping....

```

columns(org.Hs.eg.db)

```

```

[1] "ACCCNUM"       "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 can use the `mapIds()` function now to “translate” between any of these databases

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`.

The mapIds() function was used to add Entrez IDs, UniProt accessions, and gene names to the DESeq2 results table.

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

'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 7 columns
  baseMean log2FoldChange     lfcSE      stat    pvalue
  <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG000000000003 747.194195 -0.350703  0.168242 -2.084514 0.0371134
ENSG000000000005  0.000000      NA       NA       NA       NA
ENSG000000000419 520.134160  0.206107  0.101042  2.039828 0.0413675
ENSG000000000457 322.664844  0.024527  0.145134  0.168996 0.8658000
ENSG000000000460 87.682625 -0.147143  0.256995 -0.572550 0.5669497
ENSG000000000938 0.319167   -1.732289  3.493601 -0.495846 0.6200029
  padj      symbol
  <numeric> <character>
ENSG000000000003 0.163017    TSPAN6
ENSG000000000005  NA          TNMD
ENSG000000000419 0.175937    DPM1
ENSG000000000457 0.961682    SCYL3
ENSG000000000460 0.815805    FIRRM
ENSG000000000938  NA          FGR
```

Q. Also add “ENTREZID”, “GENENAME” IDs to our `res` object

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

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

```

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

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

```

You can arrange and view the results by the adjusted p-value

```

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 9 columns
  baseMean log2FoldChange      lfcSE      stat      pvalue
  <numeric>      <numeric> <numeric> <numeric>    <numeric>
ENSG00000152583   954.771      4.36836  0.2371306   18.4217 8.79214e-76
ENSG00000179094   743.253      2.86389  0.1755659   16.3123 8.06568e-60
ENSG00000116584  2277.913     -1.03470  0.0650826  -15.8983 6.51317e-57
ENSG00000189221  2383.754      3.34154  0.2124091   15.7316 9.17960e-56
ENSG00000120129  3440.704      2.96521  0.2036978   14.5569 5.27883e-48
ENSG00000148175  13493.920     1.42717  0.1003811   14.2175 7.13625e-46
  padj      symbol      entrez      genename
  <numeric> <character> <character>    <character>
ENSG00000152583 1.33157e-71    SPARCL1      8404      SPARC like 1
ENSG00000179094 6.10774e-56    PER1        5187 period circadian reg..
ENSG00000116584 3.28806e-53   ARHGEF2      9181 Rho/Rac guanine nucl..
ENSG00000189221 3.47563e-52    MAOA        4128 monoamine oxidase A
ENSG00000120129 1.59896e-44   DUSP1        1843 dual specificity pho..
ENSG00000148175 1.80131e-42    STOM        2040      stomatin

```

Save annotated results to a CSV file

```

write.csv(res, file = "results_annotated.csv")

```

Volcano plot

```
library(EnhancedVolcano)
```

Loading required package: ggrepel

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

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

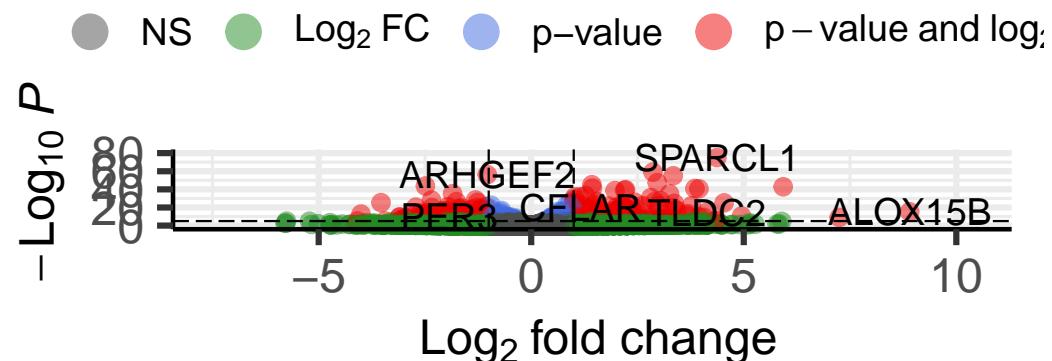
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.

i The deprecated feature was likely used in the EnhancedVolcano package.
Please report the issue to the authors.

Warning: The `size` argument of `element_line()` is deprecated as of ggplot2 3.4.0.
i Please use the `linewidth` argument instead.
i The deprecated feature was likely used in the EnhancedVolcano package.
Please report the issue to the authors.

Volcano plot

EnhancedVolcano



total = 38694 variables

Pathway analysis

What known biological pathways do our differently expressed genes overlap with (I.e. play role in?)

There is lots of bioconductor packages to do this type of analysis.

We will use one of the oldest called **gage** along with **pathview** to render nice pics of the pathways we find

we can install these with the command `BiocManager::install(c("pathview", "gage", "gageData"))`

```
library(pathview)
library(gage)
library(gageData)
```

have a wee peak what is in `gageData`

```
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"
```

The main `gage()` function that does the work wants a simple vector as input.

```
foldchanges <- res$log2FoldChange
names(foldchanges) <- res$symbol
head(foldchanges)
```

TSPAN6	TNMD	DPM1	SCYL3	FIRRM	FGR
-0.35070296	NA	0.20610728	0.02452701	-0.14714263	-1.73228897

The KEGG database uses ENTREZ ids so we need to provide these in our input vector for `gage`

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

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

What is in the output object `keggres`

```
attributes(keggres)
```

```
$names
```

```
[1] "greater" "less"     "stats"
```

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

		p.geomean	stat.mean	p.val
hsa05332	Graft-versus-host disease	0.0004250607	-3.473335	0.0004250607
hsa04940	Type I diabetes mellitus	0.0017820379	-3.002350	0.0017820379
hsa05310	Asthma	0.0020046180	-3.009045	0.0020046180

		q.val	set.size	exp1
hsa05332	Graft-versus-host disease	0.09053792	40	0.0004250607
hsa04940	Type I diabetes mellitus	0.14232788	42	0.0017820379
hsa05310	Asthma	0.14232788	29	0.0020046180

We can use the `pathview` function to render a figure of any of these pathways along with annotation for our DEGs.

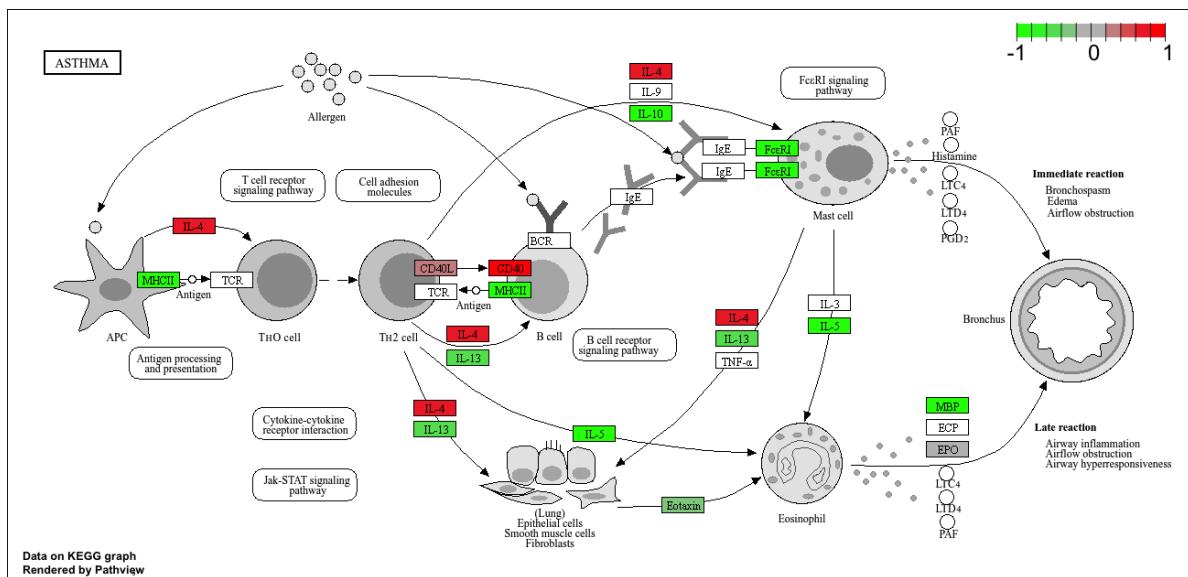
Lets now see the hsa05310 Asthma pathway with our DEGs colored up:

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

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

```
Info: Working in directory /Users/mel/Documents/BIMM143/Class13
```

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



Q. can yo render and insert here the pathway figure for “Graft-versus-host disease” and “Type I diabetes”

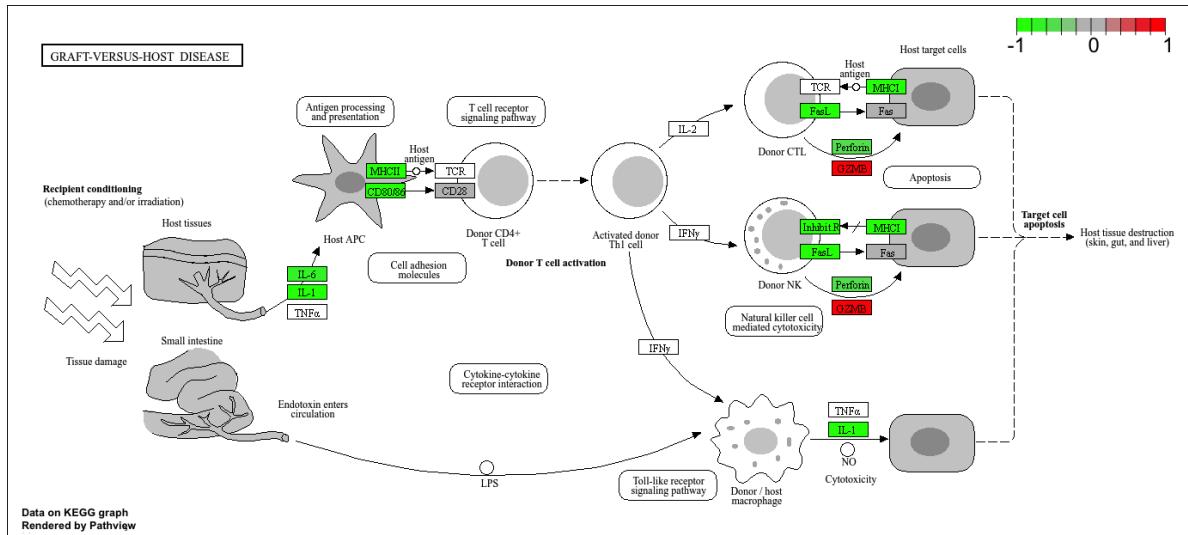
Q12. Can you do the same procedure as above to plot the pathview figures for the top 2 down-regulated pathways?

```
pathview(gene.data=foldchanges, pathway.id="hsa05332")
```

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

```
Info: Working in directory /Users/mel/Documents/BIMM143/Class13
```

```
Info: Writing image file hsa05332.pathview.png
```



```
pathview(gene.data=foldchanges, pathway.id="hsa04940")
```

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

```
Info: Working in directory /Users/mel/Documents/BIMM143/Class13
```

```
Info: Writing image file hsa04940.pathview.png
```

