

Class13: Transcriptomics and the analysis of RNA-Seq data

Yi-Hung Lee (A16587141)

The data for this hands-on session comes from a published RNA-seq experiment where airway smooth muscle cells were treated with dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects

Import data

Load library DESeq2

```
library(BiocManager)
```

```
Bioconductor version '3.18' is out-of-date; the current release version '3.19'  
is available with R version '4.4'; see https://bioconductor.org/install
```

```
library(DESeq2)
```

```
Warning: package 'DESeq2' was built under R version 4.3.3
```

```
Loading required package: S4Vectors
```

```
Loading required package: stats4
```

```
Loading required package: BiocGenerics
```

```
Attaching package: 'BiocGenerics'
```

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

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

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

```
anyDuplicated, aperm, append, as.data.frame, basename, cbind,
colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
table, tapply, union, unique, unsplit, which.max, which.min
```

```
Attaching package: 'S4Vectors'
```

```
The following 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
```

```
Warning: package 'GenomeInfoDb' was built under R version 4.3.3
```

```
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
```

```
Load data
```

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

```
head(counts)
```

	SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516
ENSG00000000003	723	486	904	445	1170
ENSG00000000005	0	0	0	0	0
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		
ENSG00000000003	1097	806	604		
ENSG00000000005	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?

There are 38694 genes in this dataset

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

There are 4 control cell lines

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

```
[1] 4
```

Check on match of the dataset

```
colnames(counts)

[1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"
[6] "SRR1039517" "SRR1039520" "SRR1039521"

metadata$id

[1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"
[6] "SRR1039517" "SRR1039520" "SRR1039521"

colnames(counts) == metadata$id

[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

## Data matched

## See if they are all true

all(colnames(counts) == metadata$id)

[1] TRUE
```

Analysis

Wanna start comparing ‘control’ to ‘treated’ columns. To do this I will find the mean of the ‘control’ column.

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

ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         900.75          0.00        520.50        339.75        97.25
ENSG000000000938
         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)

```
treated <- metadata[metadata[, "dex"] == 'treated', ]  
treated.count <- counts[, treated$id]  
treated.mean <- apply(treated.count, 1, mean)  
head(treated.mean)
```

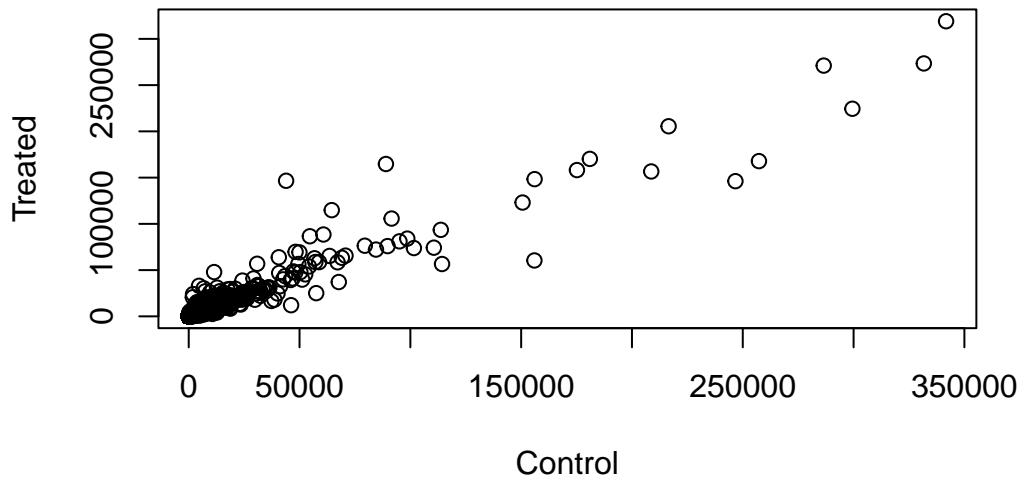
```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460  
658.00          0.00          546.00         316.50          78.75  
ENSG00000000938  
0.00
```

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

	control.mean	treated.mean
ENSG00000000003	900.75	658.00
ENSG00000000005	0.00	0.00
ENSG00000000419	520.50	546.00
ENSG00000000457	339.75	316.50
ENSG00000000460	97.25	78.75
ENSG00000000938	0.75	0.00

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.

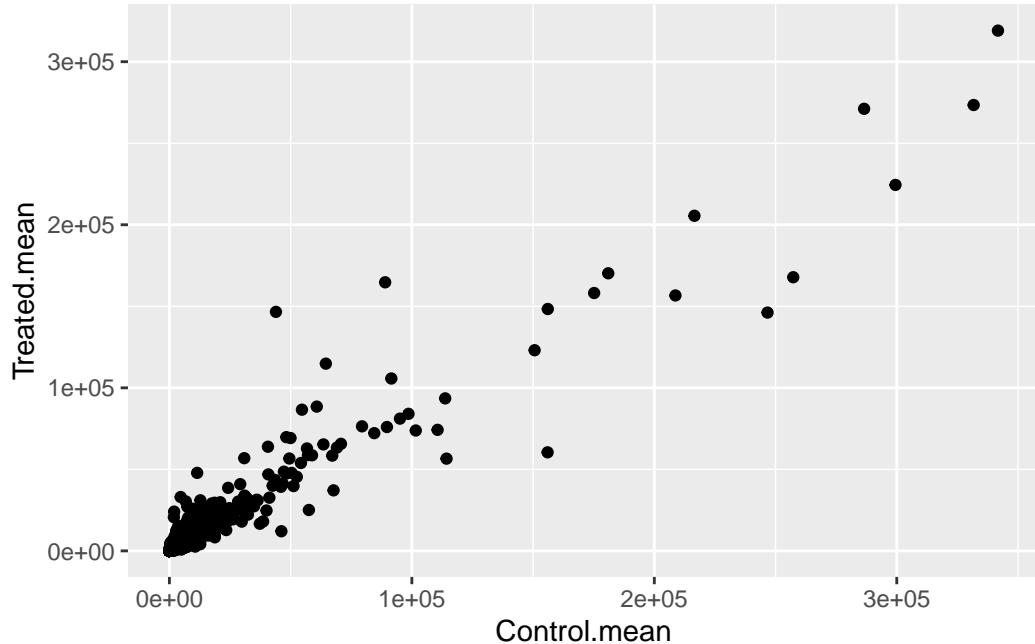
```
plot(meancounts[,1], meancounts[, 2], xlab = "Control", ylab = 'Treated')
```



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(meancounts[,1], meancounts[, 2])) +
  geom_point() +
  xlab("Control.mean") +
  ylab('Treated.mean')
```

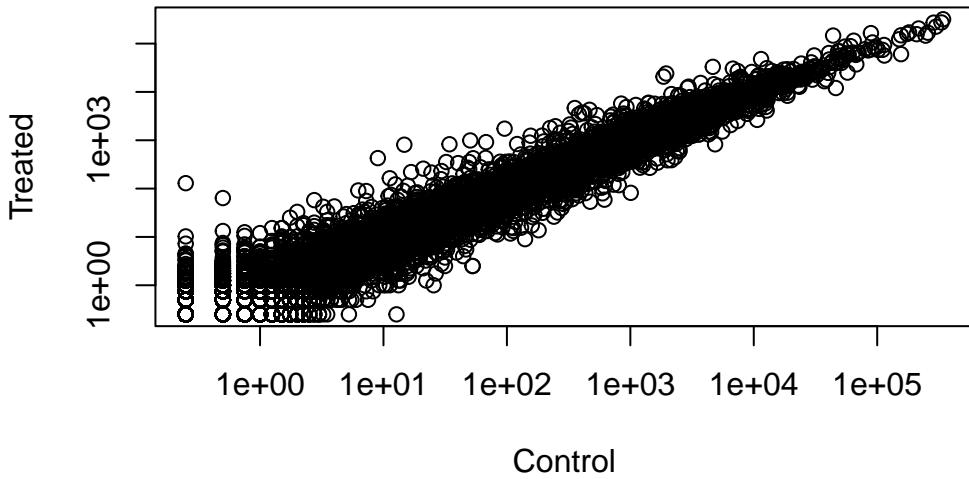


Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

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

Warning in xy.coords(x, y, xlabel, ylabel, log): 15032 x values <= 0 omitted from logarithmic plot

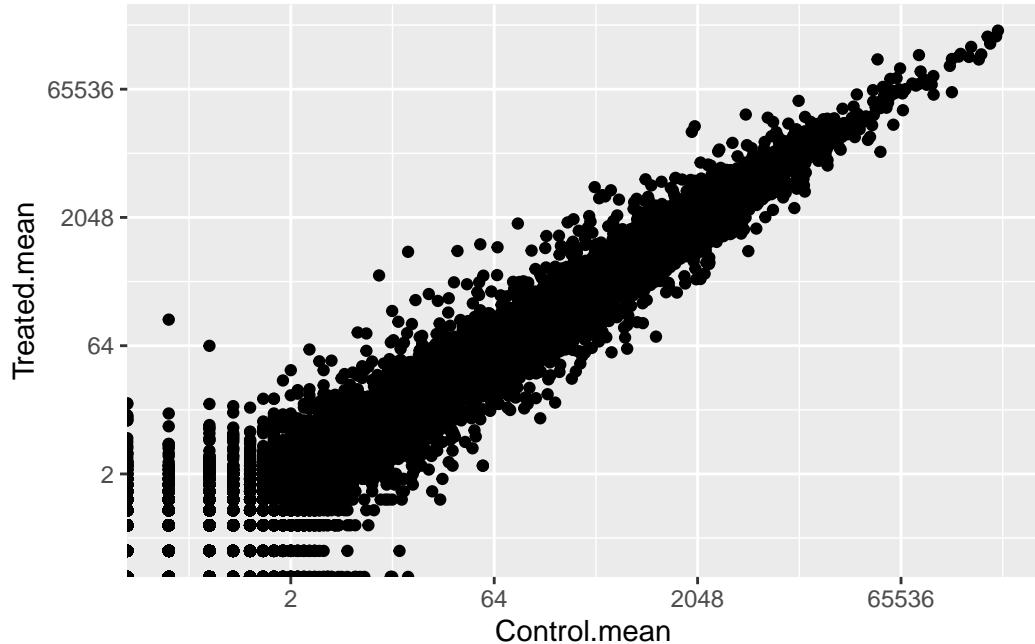
Warning in xy.coords(x, y, xlabel, ylabel, log): 15281 y values <= 0 omitted from logarithmic plot



```
ggplot(meancounts, aes(meancounts[,1], meancounts[, 2])) +  
  geom_point() +  
  xlab("Control.mean") +  
  ylab('Treated.mean') +  
  scale_x_continuous(trans="log2") +  
  scale_y_continuous(trans="log2")
```

Warning in scale_x_continuous(trans = "log2"): log-2 transformation introduced infinite values.

Warning in scale_y_continuous(trans = "log2"): log-2 transformation introduced infinite values.



```
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

Remove the datasets that includes 'NaN' or '-Inf'

```
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
--	--------------	--------------	--------

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
ENSG00000001036	2327.00	1785.75 -0.38194109

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

The arr.ind=TRUE argument will clause which() to return both the row and column indices (i.e. positions) where there are TRUE values.

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

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

```
sum(up.ind == TRUE)
```

[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?

```
sum(down.ind == TRUE)
```

[1] 367

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

No because we haven't check if the number is statistically significant yet

Setting up for DESeq

```
library(DESeq2)
citation("DESeq2")
```

To cite package 'DESeq2' in publications use:

Love, M.I., Huber, W., Anders, S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2 Genome Biology 15(12):550 (2014)

A BibTeX entry for LaTeX users is

```
@Article{,
  title = {Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2},
  author = {Michael I. Love and Wolfgang Huber and Simon Anders},
  year = {2014},
  journal = {Genome Biology},
  doi = {10.1186/s13059-014-0550-8},
  volume = {15},
  issue = {12},
  pages = {550},
}
```

Import Data

```
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

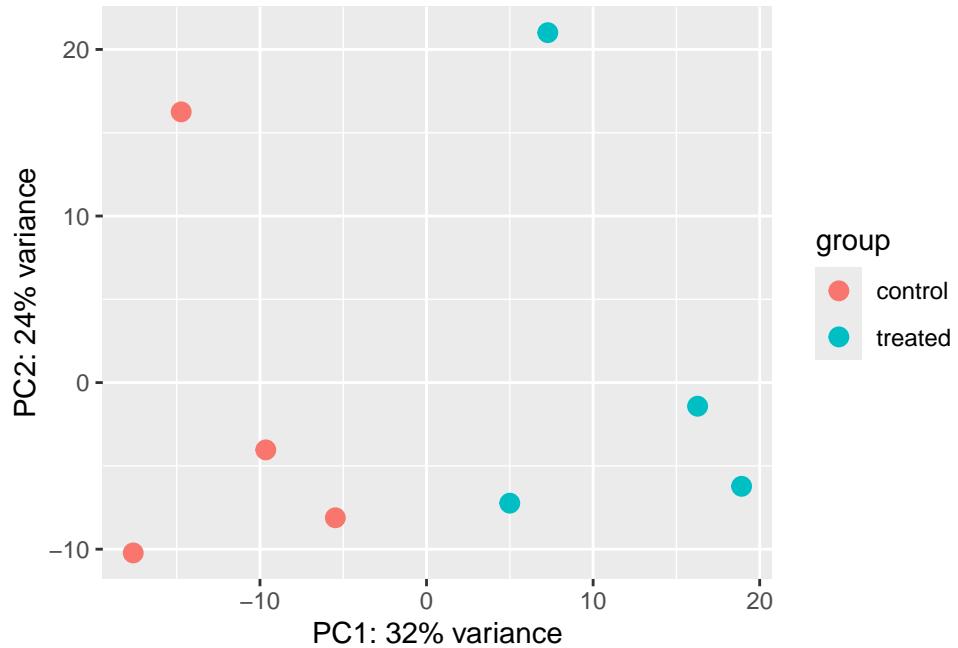
```
dds
```

```
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
```

Principal Component Analysis (PCA)

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

using ntop=500 top features by variance



Use ggplot to plot PCA

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

using ntop=500 top features by variance

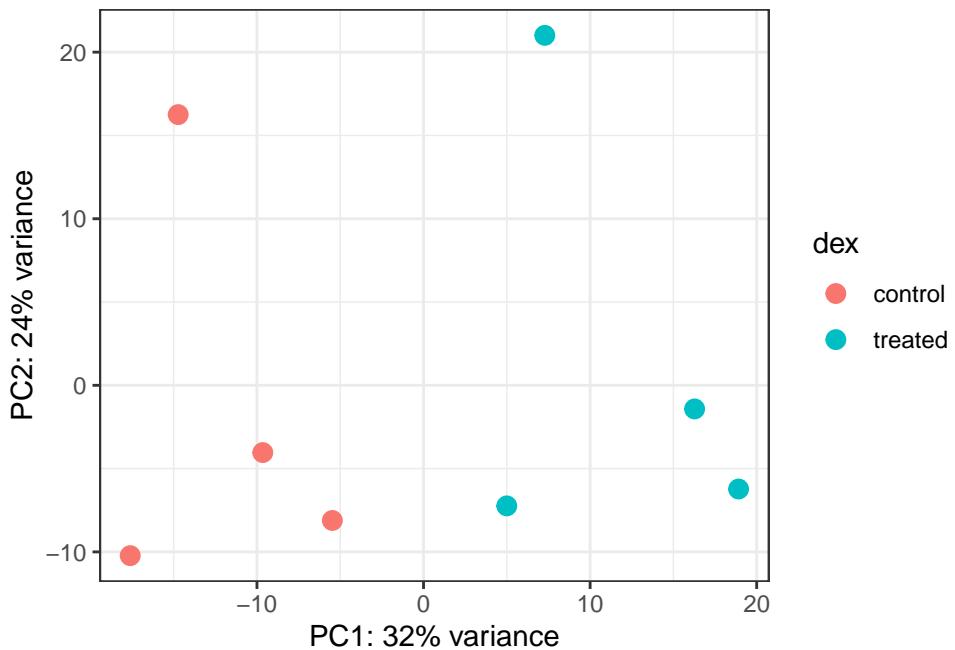
```
head(pcaData)
```

	PC1	PC2	group	dex	name
SRR1039508	-17.607922	-10.225252	control	control	SRR1039508
SRR1039509	4.996738	-7.238117	treated	treated	SRR1039509

```
SRR1039512 -5.474456 -8.113993 control control SRR1039512
SRR1039513 18.912974 -6.226041 treated treated SRR1039513
SRR1039516 -14.729173 16.252000 control control SRR1039516
SRR1039517 7.279863 21.008034 treated treated SRR1039517
```

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

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



DESeq analysis

```
dds <- DESeq(dds)

estimating size factors
```

```

estimating dispersions

gene-wise dispersion estimates

mean-dispersion relationship

final dispersion estimates

fitting model and testing

```

```

res <- results(dds)
res

```

```

log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 38694 rows and 6 columns
  baseMean log2FoldChange    lfcSE     stat   pvalue
  <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG000000000003  747.1942 -0.3507030  0.168246 -2.084470 0.0371175
ENSG000000000005   0.0000      NA       NA       NA       NA
ENSG000000000419  520.1342  0.2061078  0.101059  2.039475 0.0414026
ENSG000000000457  322.6648  0.0245269  0.145145  0.168982 0.8658106
ENSG000000000460   87.6826 -0.1471420  0.257007 -0.572521 0.5669691
...
...
ENSG00000283115  0.000000      NA       NA       NA       NA
ENSG00000283116  0.000000      NA       NA       NA       NA
ENSG00000283119  0.000000      NA       NA       NA       NA
ENSG00000283120  0.974916 -0.668258  1.69456 -0.394354 0.693319
ENSG00000283123  0.000000      NA       NA       NA       NA
  padj
  <numeric>
ENSG000000000003  0.163035
ENSG000000000005      NA
ENSG000000000419  0.176032
ENSG000000000457  0.961694
ENSG000000000460  0.815849
...
...
ENSG00000283115      NA
ENSG00000283116      NA
ENSG00000283119      NA
ENSG00000283120      NA
ENSG00000283123      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
```

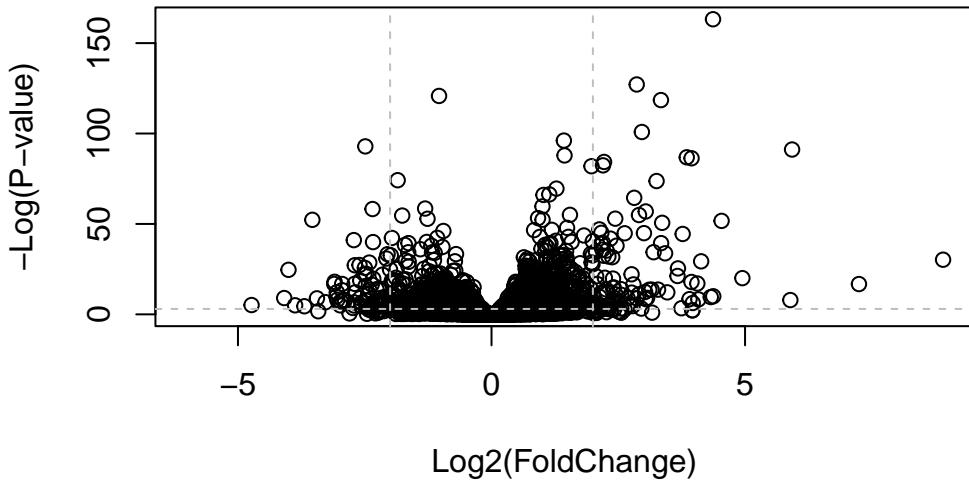
Set cutoff to 0.05

```
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
```

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

# Cut-off lines
abline(v=c(-2,2), col="gray", lty=2)
abline(h=-log(0.05), col="gray", lty=2)
```



```

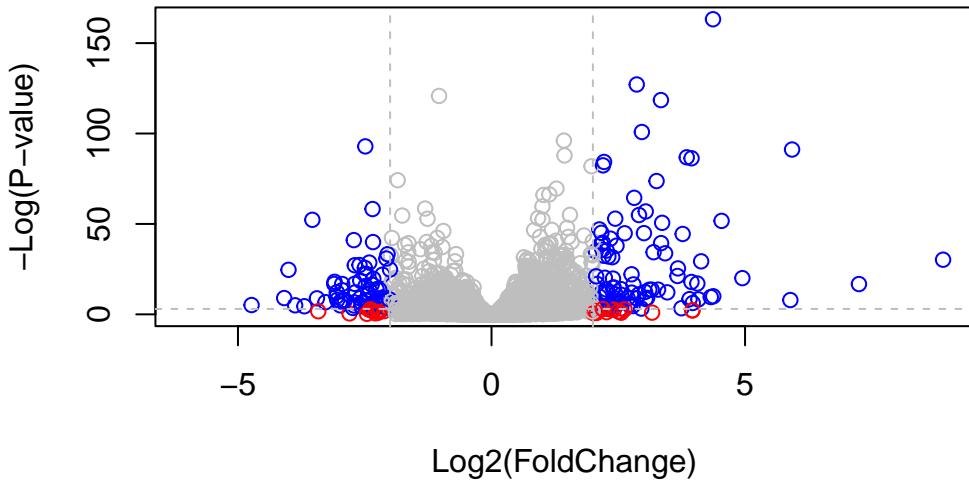
mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ] <- "red"

inds <- (res$padj < 0.05) & (abs(res$log2FoldChange) > 2 )
mycols[ inds ] <- "blue"

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

# Cut-off lines
abline(v=c(-2,2), col="gray", lty=2)
abline(h=-log(0.05), col="gray", lty=2)

```



```

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

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
                      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 7 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
ENSG000000000419	520.134160	0.2061078	0.101059	2.039475	0.0414026
ENSG000000000457	322.664844	0.0245269	0.145145	0.168982	0.8658106
ENSG000000000460	87.682625	-0.1471420	0.257007	-0.572521	0.5669691
ENSG000000000938	0.319167	-1.7322890	3.493601	-0.495846	0.6200029
	padj	symbol			
	<numeric>	<character>			
ENSG000000000003	0.163035	TSPAN6			
ENSG000000000005	NA	TNMD			
ENSG000000000419	0.176032	DPM1			
ENSG000000000457	0.961694	SCYL3			
ENSG000000000460	0.815849	FIRRM			
ENSG000000000938	NA	FGR			