

Class 13

Gaeun Jun (A16814573)

```
library(BiocManager)
library(DESeq2)
```

Loading required package: S4Vectors

Loading required package: stats4

Loading required package: BiocGenerics

Attaching package: 'BiocGenerics'

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

IQR, mad, sd, var, xtabs

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

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

Attaching package: 'S4Vectors'

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

```
  findMatches
```

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

```
  expand.grid, I, unname
```

```
Loading required package: IRanges
```

```
Loading required package: GenomicRanges
```

```
Loading required package: GenomeInfoDb
```

```
Loading required package: SummarizedExperiment
```

```
Loading required package: MatrixGenerics
```

```
Loading required package: matrixStats
```

```
Attaching package: 'MatrixGenerics'
```

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

```
  colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
  colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
  colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
  colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
  colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
  colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
  colWeightedMeans, colWeightedMedians, colWeightedSds,
  colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
  rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
  rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
  rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
  rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
  rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
  rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
  rowWeightedSds, rowWeightedVars
```

```
Loading required package: Biobase
```

```
Welcome to Bioconductor
```

```
Vignettes contain introductory material; view with  
'browseVignettes()'. To cite Bioconductor, see  
'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
Attaching package: 'Biobase'
```

```
The following object is masked from 'package:MatrixGenerics':
```

```
rowMedians
```

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

```
anyMissing, rowMedians
```

3. Import countData and colData

```
counts <- read.csv("https://bioboot.github.io/bimm143_W18/class-material/airway_scaledcounts  
metadata <- read.csv("https://bioboot.github.io/bimm143_W18/class-material/airway_metadata.c
```

```
head(counts)
```

	SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516
ENSG000000000003	723	486	904	445	1170
ENSG000000000005	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		
ENSG000000000003	1097	806	604		
ENSG000000000005	0	0	0		
ENSG00000000419	781	417	509		
ENSG00000000457	447	330	324		
ENSG00000000460	94	102	74		
ENSG00000000938	0	0	0		

```
head(metadata)
```

```
      id      dex celltype      geo_id
1 SRR1039508 control    N61311 GSM1275862
2 SRR1039509 treated    N61311 GSM1275863
3 SRR1039512 control    N052611 GSM1275866
4 SRR1039513 treated    N052611 GSM1275867
5 SRR1039516 control    N080611 GSM1275870
6 SRR1039517 treated    N080611 GSM1275871
```

Sanity check on correspondence of counts and metadata:

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

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

```
View(counts)
```

There are 38694 genes in this dataset

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

```
metadata$dex
```

```
[1] "control" "treated" "control" "treated" "control" "treated" "control"
[8] "treated"
```

```
n.control <- sum(metadata$dex == "control")
```

There are 4 ‘control’ cell lines in this dataset.

4. Toy differential gene expression

Finding sample id for the labeled control and calculating the mean counts per gene across the samples.

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

```
ENSG00000000003 ENSG00000000005 ENSG000000000419 ENSG000000000457 ENSG000000000460  
900.75          0.00        520.50        339.75        97.25  
ENSG000000000938  
0.75
```

```
library(dplyr)
```

Attaching package: 'dplyr'

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

```
combine
```

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

```
count
```

The following objects are masked from 'package:GenomicRanges':

```
intersect, setdiff, union
```

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

```
intersect
```

The following objects are masked from 'package:IRanges':

```
collapse, desc, intersect, setdiff, slice, union
```

```
The following objects are masked from 'package:S4Vectors':
```

```
  first, intersect, rename, setdiff, setequal, union
```

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

```
  combine, intersect, setdiff, union
```

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

```
  filter, lag
```

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

```
  intersect, setdiff, setequal, union
```

```
control <- metadata %>% filter(dex=="control")
control.counts <- counts %>% select(control$id)
control.mean <- rowSums(control.counts)/4
head(control.mean)
```

ENSG00000000003	ENSG00000000005	ENSG000000000419	ENSG000000000457	ENSG000000000460
900.75	0.00	520.50	339.75	97.25
ENSG00000000938				
0.75				

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

Instead of using `rowSums` and dividing the sample by 4 (by knowing how many control samples there are), you can use the `rowMeans` function. This is more robust because when we add samples to our data, we need to update our `rowSums` function while, `rowMeans` can automatically run through that update.

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

```

ENSG000000000003 ENSG000000000005 ENSG000000000419 ENSG000000000457 ENSG000000000460
    658.00          0.00      546.00      316.50      78.75
ENSG000000000938
    0.00

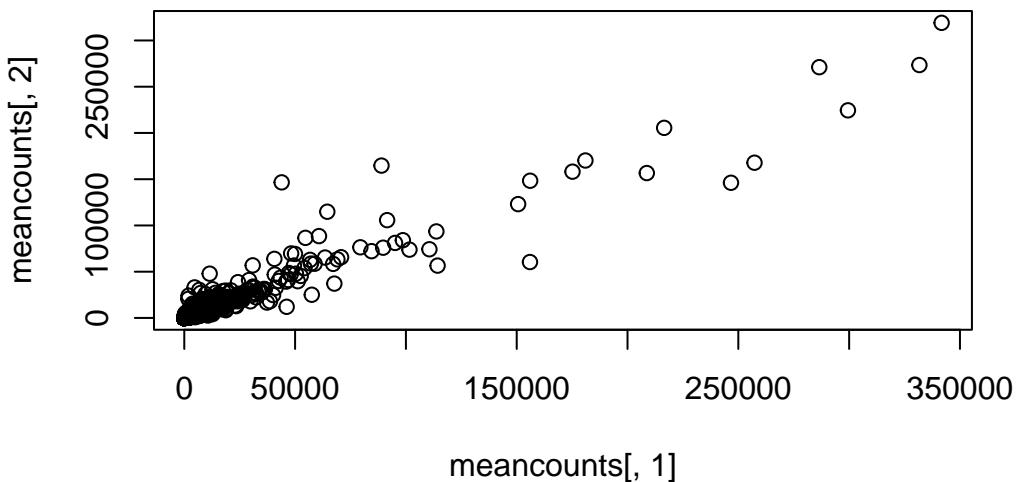
```

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

Let's make a plot to explore the results a little:

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])
```



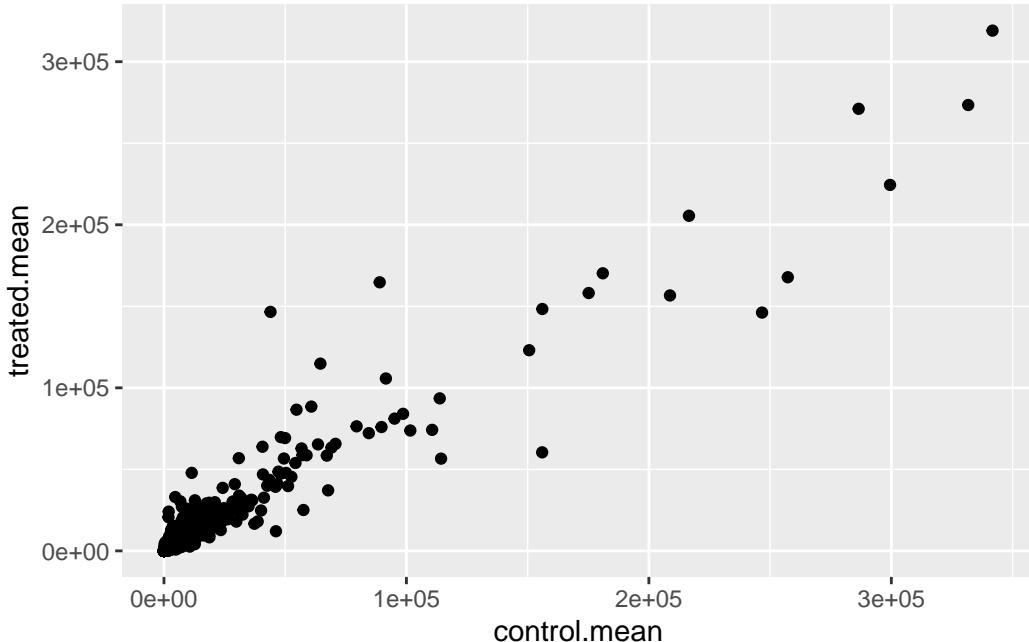
Q5 (b). You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot?

```

library(ggplot2)

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

```



You would use the geom_point function for this plot.

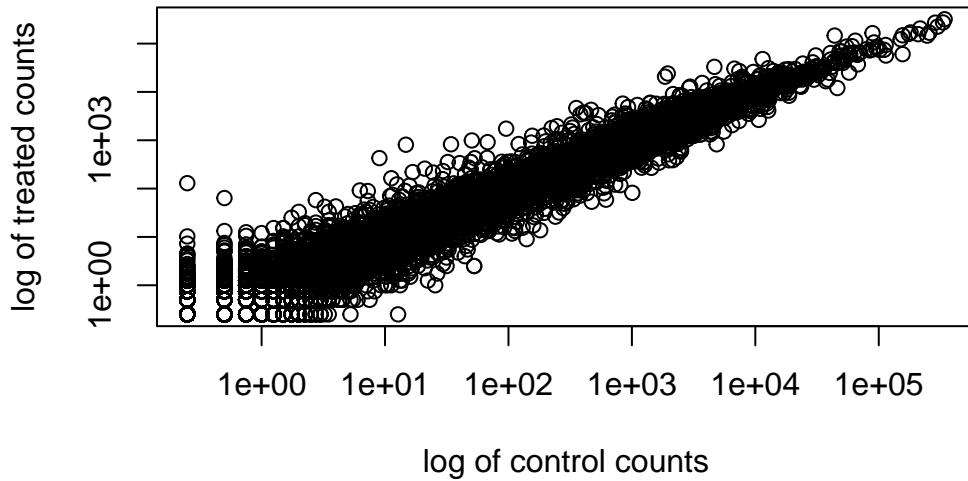
Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

We will make a log-log plot to draw out this skewed data and see what is going on.

```
plot(meancounts[,1], meancounts[,2], log="xy",
     xlab="log of control counts",
     ylab="log of treated counts")
```

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



The argument to `plot()` both axes on a log scale is `log="xy"`.

We often use `log2` transformations when dealing with this sort of data.

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

```
[1] 0
```

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

```
[1] 1
```

This `log2` transformation has this nice property where if there is no change the `log2` value will be zero and if it double the `log2` value will be 1 and if halved it will be -1.

So lets add a `log2` fold change column to our results so far:

```
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 get rid of zero count genes that we can not say anything about

```
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 argument in the which() function returns any values in the rows or columns that are TRUE. It will tell us which rows have zero counts so that we can ignore them. We then take the first column of the output and call the unique() function so that we don't accidentally count rows that have zero values twice.

How many genes are remaining?

```
nrow(mycounts)
```

[1] 21817

A common threshold used for calling something differentially expressed is a log2(FoldChange) of greater than 2 or less than -2. Let's filter the dataset both ways to see how many genes are up or down-regulated.

```
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)
```

```
[1] 250
```

There are 250 up regulated genes at the greater than 2 fc level.

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)
```

```
[1] 367
```

There are 367 down regulated genes at the greater than 2 fc level.

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

I don't entirely trust these results because it doesn't tell us anything about the significance of the fold change. These data could be misleading.

5. Setting up for DESeq

Let's do this the right way. DESeq2 is an R package specifically for analyzing count-based NGS data like RNA-seq. It is available from Bioconductor. Bioconductor is a project to provide tools for analyzing high-throughput genomic data including RNA-seq, ChIP-seq and arrays.

```
# load up DESeq2  
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},
}
```

```
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
```

Running DESeq pipeline on the dds object.

```
dds <- DESeq(dds)

estimating size factors

estimating dispersions

gene-wise dispersion estimates

mean-dispersion relationship

final dispersion estimates

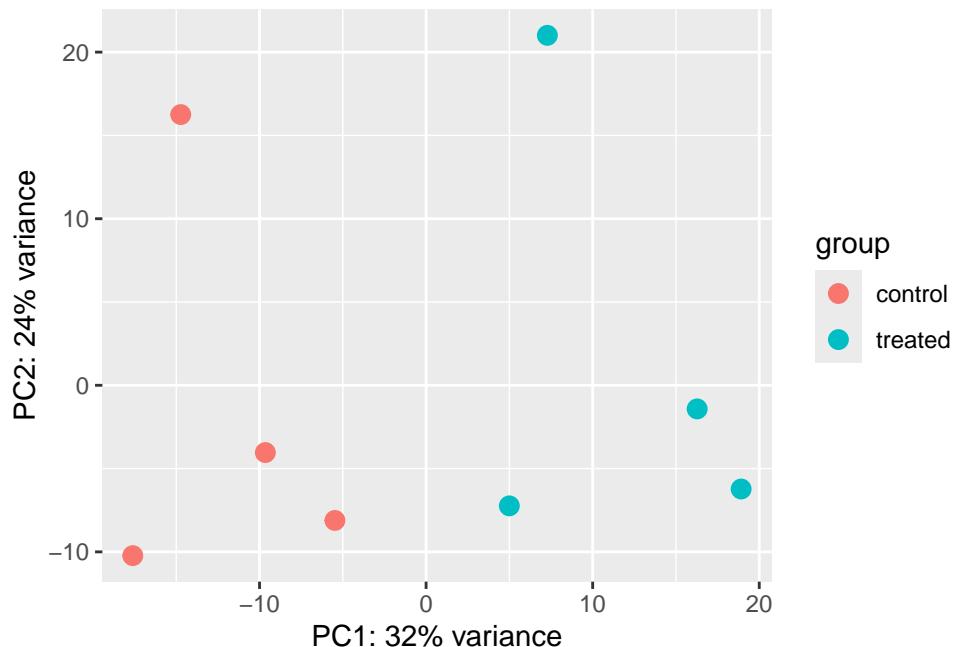
fitting model and testing
```

6. Principal Component Analysis (PCA)

Calculating and plotting PCs.

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

using ntop=500 top features by variance



Build PCA from scratch using ggplot2 package.

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

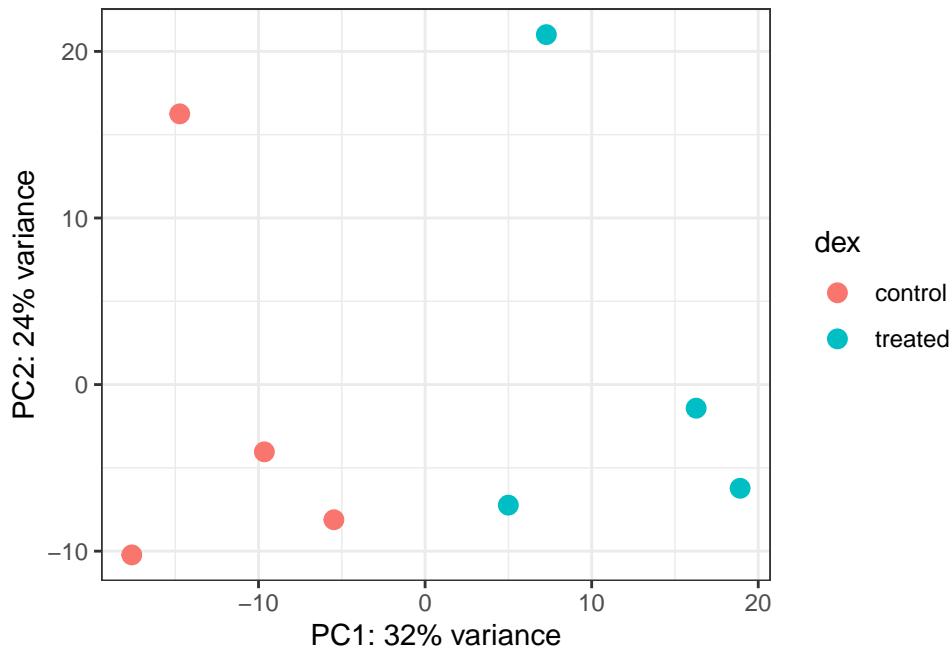
using ntop=500 top features by variance

```
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

```
# 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()
```



7. DESeq Analysis

```
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
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
ENSG00000283116	0.000000		NA	NA	NA
ENSG00000283119	0.000000		NA	NA	NA
ENSG00000283120	0.974916	-0.668258	1.69456	-0.394354	0.693319
ENSG00000283123	0.000000		NA	NA	NA

```

      padj
      <numeric>
ENSG000000000003  0.163035
ENSG000000000005      NA
ENSG000000000419  0.176032
ENSG000000000457  0.961694
ENSG000000000460  0.815849
...
      ...
ENSG0000283115      NA
ENSG0000283116      NA
ENSG0000283119      NA
ENSG0000283120      NA
ENSG0000283123      NA

```

We can get some basic summary tallies using the `summary()` function

```
summary(res, alpha = 0.5)
```

```

out of 25258 with nonzero total read count
adjusted p-value < 0.5
LFC > 0 (up)      : 3401, 13%
LFC < 0 (down)     : 3117, 12%
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

```

8. Adding annotation data

Alternative genes names and annotation for informative interpretation of results.

```
library("AnnotationDbi")
```

```
Attaching package: 'AnnotationDbi'
```

```
The following object is masked from 'package:dplyr':
```

```
select
```

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

Getting a list of available key types to map in the organism annotation package:

```
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"
```

Using `mapIds()` function to add individual columns to 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  
                      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  
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
```

```

<numeric> <character>
ENSG000000000003 0.163035      TSPAN6
ENSG000000000005 NA          TNMD
ENSG000000000419 0.176032      DPM1
ENSG000000000457 0.961694      SCYL3
ENSG000000000460 0.815849      FIRRM
ENSG000000000938 NA          FGR

```

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.

```

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

```

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

```

res$uniprot <- mapIds(org.Hs.eg.db,
                      keys=row.names(res),
                      column="UNIPROT",
                      keytype="ENSEMBL",
                      multiVals="first")

```

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

```

res$genename <- mapIds(org.Hs.eg.db,
                      keys=row.names(res),
                      column="GENENAME",
                      keytype="ENSEMBL",
                      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 10 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     uniprot
      <numeric> <character> <character> <character>
ENSG000000000003 0.163035   TSPAN6      7105 AOA087WYV6
ENSG000000000005  NA        TNMD      64102 Q9H2S6
ENSG00000000419 0.176032   DPM1      8813 H0Y368
ENSG00000000457 0.961694   SCYL3      57147 X6RHX1
ENSG00000000460 0.815849   FIRRM      55732 A6NFP1
ENSG00000000938  NA        FGR       2268 B7Z6W7
      genename
      <character>
ENSG000000000003 tetraspanin 6
ENSG000000000005 tenomodulin
ENSG00000000419 dolichyl-phosphate m..
ENSG00000000457 SCY1 like pseudokina..
ENSG00000000460 FIGNL1 interacting r..
ENSG00000000938 FGR proto-oncogene, ..

```

Arrange and view results by 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 10 columns
      baseMean log2FoldChange     lfcSE      stat    pvalue
      <numeric>     <numeric> <numeric> <numeric> <numeric>
ENSG00000152583 954.771      4.36836 0.2371268 18.4220 8.74490e-76
ENSG00000179094 743.253      2.86389 0.1755693 16.3120 8.10784e-60
ENSG00000116584 2277.913     -1.03470 0.0650984 -15.8944 6.92855e-57

```

ENSG00000189221	2383.754	3.34154	0.2124058	15.7319	9.14433e-56	
ENSG00000120129	3440.704	2.96521	0.2036951	14.5571	5.26424e-48	
ENSG00000148175	13493.920	1.42717	0.1003890	14.2164	7.25128e-46	
	padj	symbol	entrez	uniprot		
	<numeric>	<character>	<character>	<character>		
ENSG00000152583	1.32441e-71	SPARCL1	8404	B4E2Z0		
ENSG00000179094	6.13966e-56	PER1	5187	A2I2P6		
ENSG00000116584	3.49776e-53	ARHGEF2	9181	A0A8Q3SIN5		
ENSG00000189221	3.46227e-52	MAOA	4128	B4DF46		
ENSG00000120129	1.59454e-44	DUSP1	1843	B4DRR4		
ENSG00000148175	1.83034e-42	STOM	2040	F8VSL7		
	genename					
	<character>					
ENSG00000152583		SPARC like 1				
ENSG00000179094		period circadian reg..				
ENSG00000116584		Rho/Rac guanine nucl..				
ENSG00000189221		monoamine oxidase A				
ENSG00000120129		dual specificity pho..				
ENSG00000148175		stomatin				

Writing out significant results with annotations

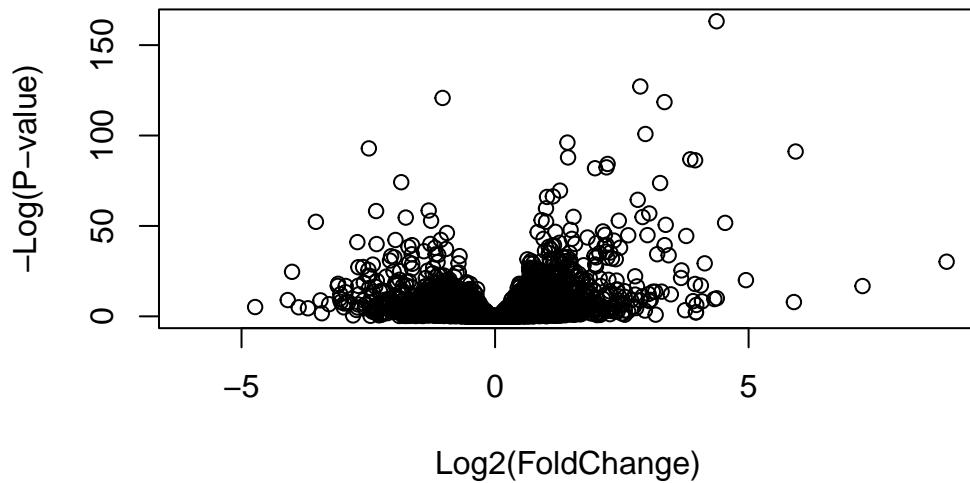
```
write.csv(res[ord,], "deseq_results.csv")
```

9. Data Visualization

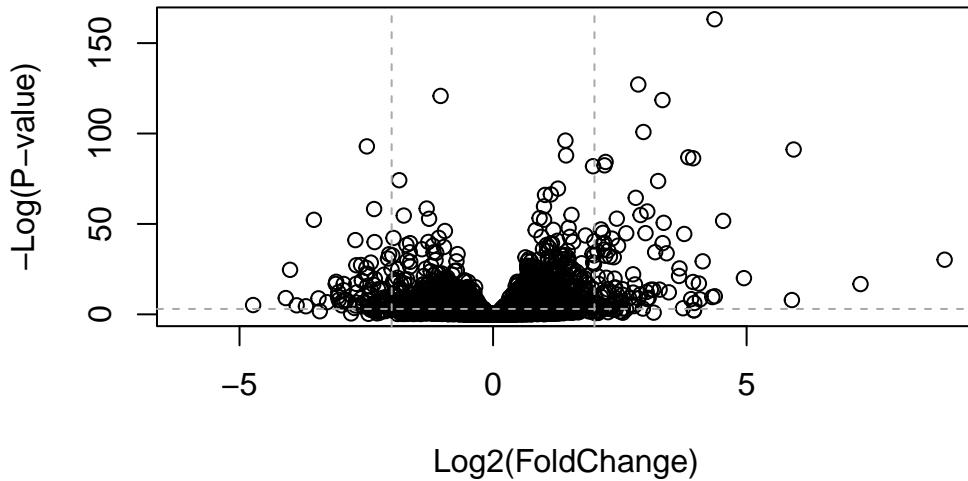
Volcano plot

Make a summary plot of our results.

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



```
plot( res$log2FoldChange, -log(res$padj),  
      ylab="-Log(P-value)", xlab="Log2(FoldChange)")  
  
# Add some cut-off lines  
abline(v=c(-2,2), col="darkgray", lty=2)  
abline(h=-log(0.05), col="darkgray", lty=2)
```



```

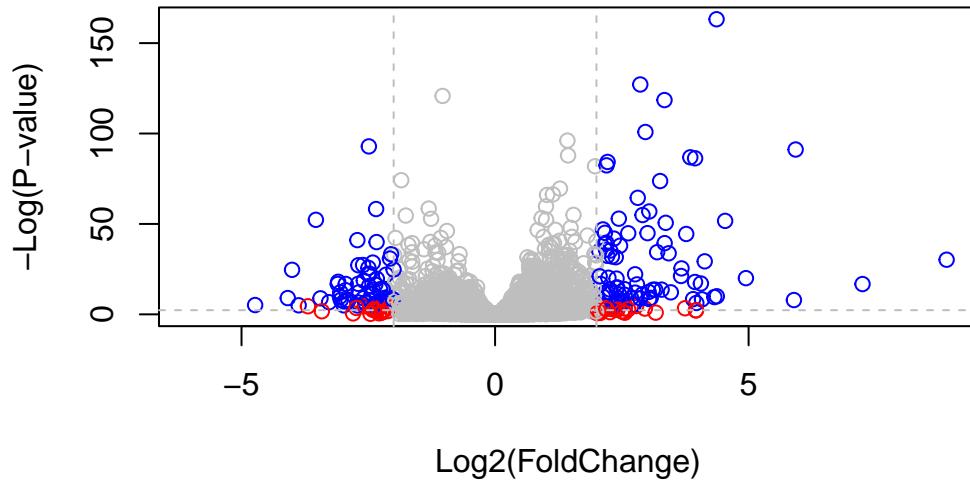
# Setup our custom point color vector
mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ] <- "red"

inds <- (res$padj < 0.01) & (abs(res$log2FoldChange) > 2 )
mycols[ inds ] <- "blue"

# Volcano plot 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)

```



Making plot more customizable

```
library(EnhancedVolcano)
```

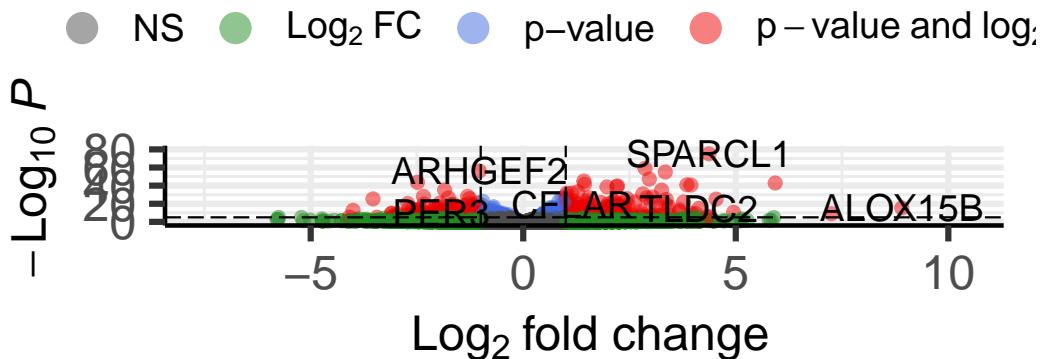
```
Loading required package: ggrepel
```

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

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

Volcano plot

Enhanced Volcano



total = 38694 variables

10. Pathway analysis

```
# loading packages
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)

# Examine the first 2 pathways in this kegg set for humans
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"

```

Fold changes for the gage() function.

```

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

Running the gage pathway analysis

```

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

```

Looking at the objects returned from gage().

```

attributes(keggres)

```

```

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

```

Looking at the first few down pathway results

```
# 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

Trying out the pathview() function

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

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

Info: Working in directory /Users/hannahjun/Desktop/bimm 143/bimm143lab13/class 13

Info: Writing image file hsa05310.pathview.png

Optional: Plotting counts for genes of interest

```
i <- grep("CRISPLD2", res$symbol)
res[i,]
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 1 row and 10 columns
  baseMean log2FoldChange      lfcSE      stat      pvalue
  <numeric>      <numeric> <numeric> <numeric>    <numeric>
ENSG00000103196  3096.16      2.62603  0.267444  9.81899 9.32747e-23
  padj      symbol      entrez      uniprot
  <numeric> <character> <character> <character>
ENSG00000103196 3.36344e-20    CRISPLD2      83716 AOA140VK80
```

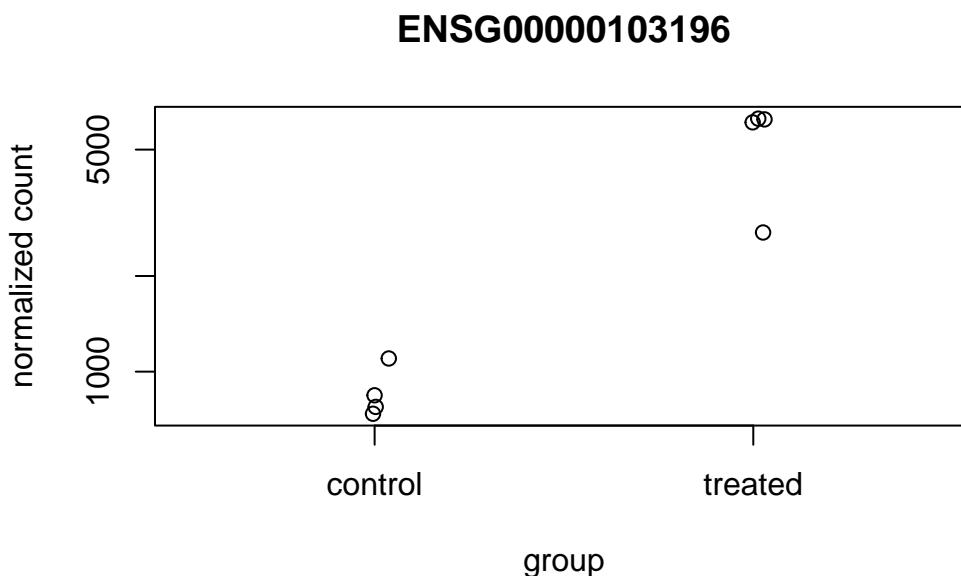
```
genename
<character>
ENSG00000103196 cysteine rich secret..
```

```
rownames(res[i,])
```

```
[1] "ENSG00000103196"
```

Plotting the counts, where the intgrup variable is the “dex” column

```
plotCounts(dds, gene="ENSG00000103196", intgroup="dex")
```

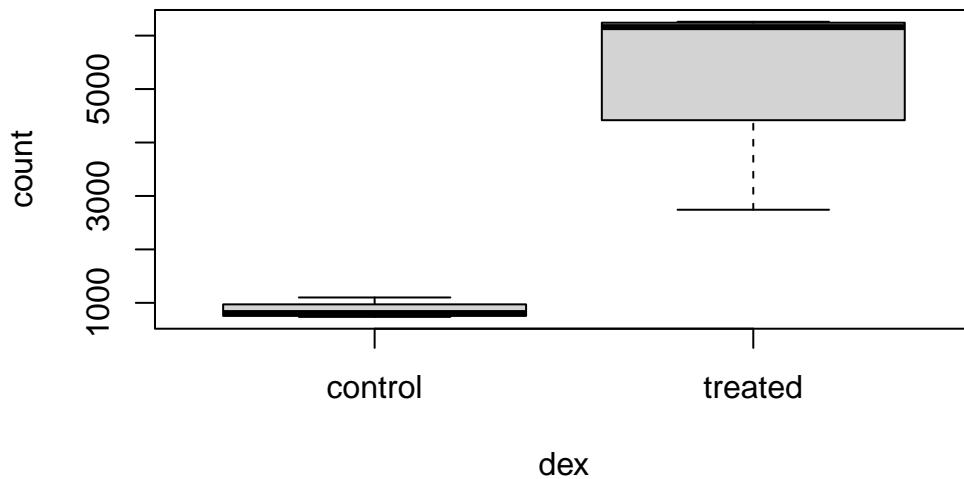


```
# Return the data
d <- plotCounts(dds, gene="ENSG00000103196", intgroup="dex", returnData=TRUE)
head(d)
```

	count	dex	group
SRR1039508	774.5002	control	
SRR1039509	6258.7915	treated	
SRR1039512	1100.2741	control	
SRR1039513	6093.0324	treated	
SRR1039516	736.9483	control	
SRR1039517	2742.1908	treated	

Making this into a boxplot

```
boxplot(count ~ dex , data=d)
```



Boxplot using ggplot2

```
library(ggplot2)
ggplot(d, aes(dex, count, fill=dex)) +
  geom_boxplot() +
  scale_y_log10() +
  ggtitle("CRISPLD2")
```

CRISPLD2

