Lab12

Karen Guerrero

```
# For this class, you'll also need DESeq2:
BiocManager::install("DESeq2")
```

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

##Download files for Lab11 We need at least two things: -count data (genes in rows and experiments in cols) - metadata (a.k.a 'colData')

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

```
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
```

```
head(counts)
```

```
                SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
ENSG00000000003        723        486        904        445       1170
ENSG00000000005          0          0          0          0          0
ENSG00000000419        467        523        616        371        582
ENSG00000000457        347        258        364        237        318
```

```
ENSG00000000460          96          81          73          66         118
ENSG00000000938           0           0           1           0           2
                SRR1039517 SRR1039520 SRR1039521
ENSG00000000003       1097         806         604
ENSG00000000005          0           0           0
ENSG00000000419        781         417         509
ENSG00000000457        447         330         324
ENSG00000000460         94         102          74
ENSG00000000938          0           0           0
```

##Quick look at the counts

```
head(counts)
```

```
                SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
ENSG00000000003        723        486        904        445       1170
ENSG00000000005          0          0          0          0          0
ENSG00000000419        467        523        616        371        582
ENSG00000000457        347        258        364        237        318
ENSG00000000460         96         81         73         66        118
ENSG00000000938          0          0          1          0          2
                SRR1039517 SRR1039520 SRR1039521
ENSG00000000003       1097         806         604
ENSG00000000005          0           0           0
ENSG00000000419        781         417         509
ENSG00000000457        447         330         324
ENSG00000000460         94         102          74
ENSG00000000938          0           0           0
```

##View Files

```
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
```

We need to make sure that the metadata (i.e. colData) and our counts match!

```
metadata$id
```

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

```
colnames(counts)
```

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

##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.

First double check the correspondence of the counts columns and the metadata rows

```
#Use the '==' test for equality. Basically is the bit on the right the same as the bit on
colnames(counts) == metadata$id
```

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

```
#Use all() function to check if all the inputs are TRUE.
all(metadata$id == colnames(counts))
```

```
[1] TRUE
```

```
#Example how if there is one not true how it will show the result
all(c(T,T,F,T))
```

```
[1] FALSE
```

How to check is everything false?

```
all(c(F, F,F))
```

```
[1] FALSE
```

##Find the mean count values per gene for control samples

```
metadata$dex == "control"
```

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

```
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 ENSG00000000419 ENSG00000000457 ENSG00000000460
         900.75            0.00          520.50          339.75           97.25
ENSG00000000938
           0.75
```

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

```
                SRR1039508 SRR1039512 SRR1039516 SRR1039520
ENSG00000000003        723        904       1170        806
ENSG00000000005          0          0          0          0
ENSG00000000419        467        616        582        417
ENSG00000000457        347        364        318        330
ENSG00000000460         96         73        118        102
ENSG00000000938          0          1          2          0
```

We want a mean value across these rows (i.e. a mean count per gene)

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

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         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?

Rather than actually writing the numeric value of samples in the code, I would state to divide by the column indicating the number of samples. (As shown below)

###control.mean <- rowSums(control.counts)/4 <- need to switch the numeric value by the data indicating the total number of samples.

##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)

See below for code:

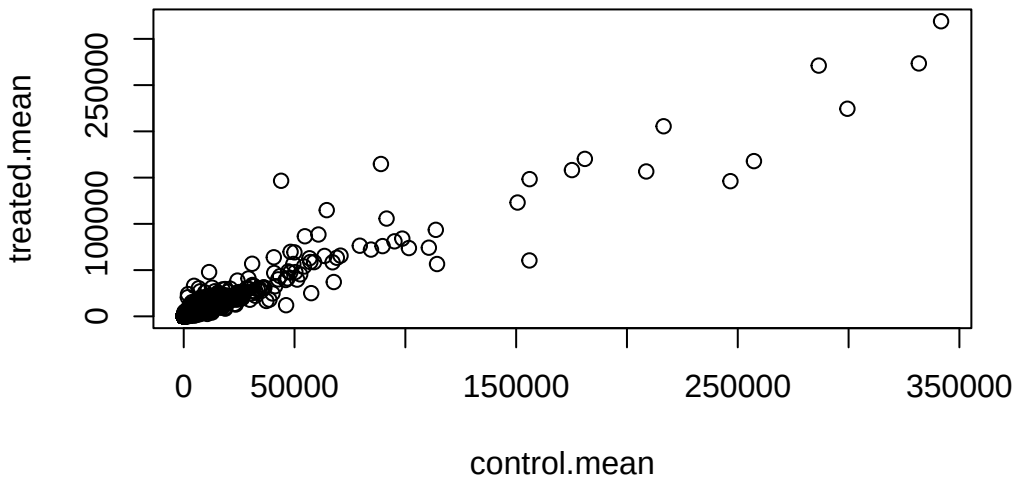
```
#Obtain the mean of treated
treated.inds <- metadata$dex == "treated"
treated.ids <- metadata[treated.inds, "id"]
treated.counts <- counts[,treated.ids]
treated.mean <- rowMeans(treated.counts)
head(treated.mean)
```

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

```
#put the means of control and treated together so we dont need to specify x-axis and y-axi
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. #Make a plot of the control.mean

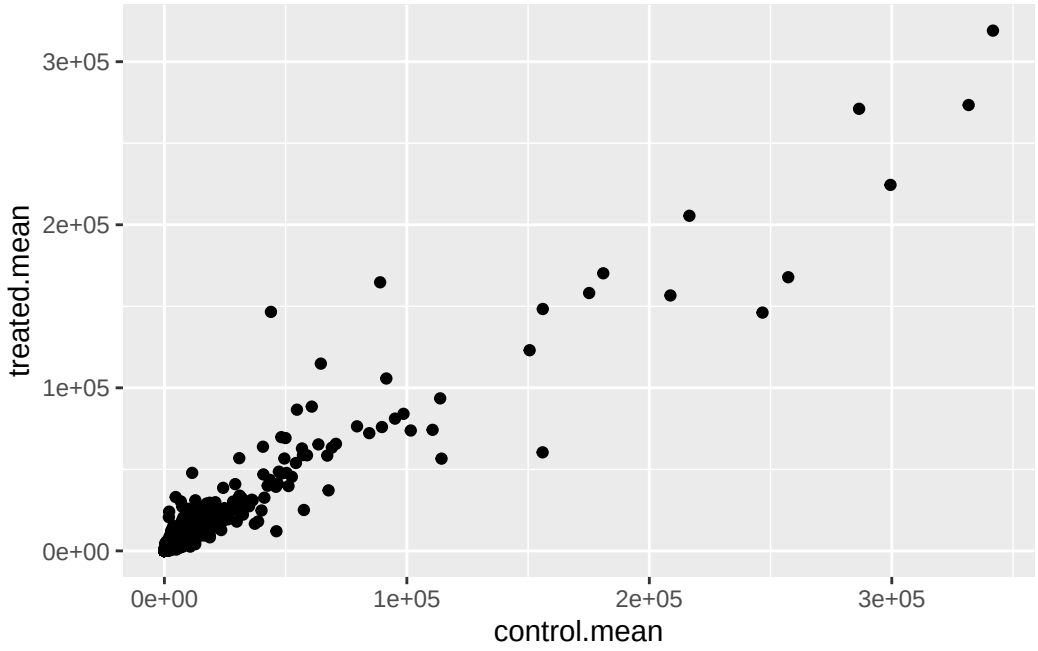
```
plot(meancounts)
```



##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?

I would use geom_point function to make plot.

```
library(ggplot2)
#Code to make plot using ggplots2
ggplot(meancounts) + aes(x=control.mean, y=treated.mean) + geom_point()
```



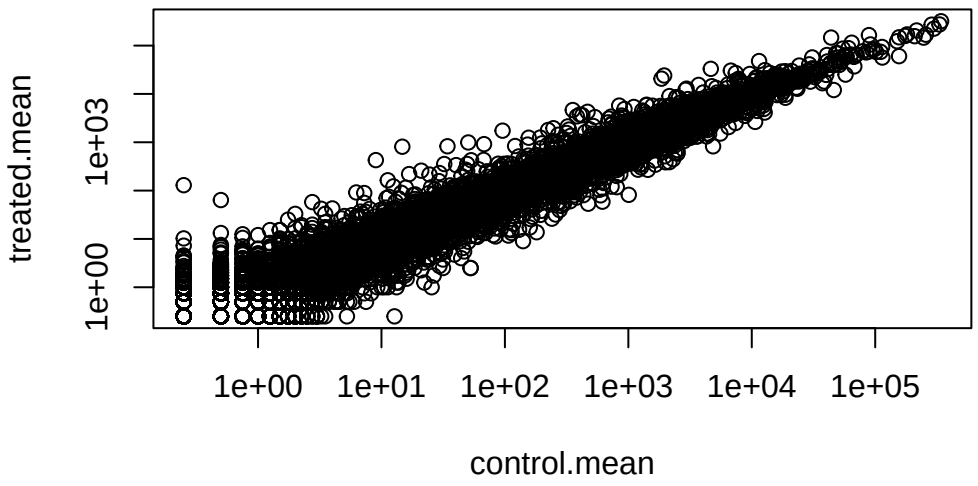
##Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

The argument is log="xy".

```
#Function to add log to plot
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
```




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

```
[1] -1
```

Log2 fold change

```
meancounts$log2fc <- log2(meancounts$treated.mean / meancounts$control.mean)
```

```
head(meancounts)
```

```
                control.mean treated.mean      log2fc
ENSG00000000003       900.75       658.00 -0.45303916
ENSG00000000005         0.00         0.00         NaN
ENSG00000000419       520.50       546.00  0.06900279
ENSG00000000457       339.75       316.50 -0.10226805
ENSG00000000460        97.25        78.75 -0.30441833
ENSG00000000938         0.75         0.00        -Inf
```

I want to get rid of any zero count genes - I can't say anything about these genes and this drug treatment anyway.

```
to.keep.inds <- rowSums(meancounts[,1:2] == 0) == 0
mycounts <- meancounts[to.keep.inds,]
head(mycounts)
```

```
                control.mean treated.mean      log2fc
ENSG00000000003       900.75       658.00 -0.45303916
ENSG00000000419       520.50       546.00  0.06900279
ENSG00000000457       339.75       316.50 -0.10226805
ENSG00000000460        97.25        78.75 -0.30441833
ENSG00000000971      5219.00      6687.50  0.35769358
ENSG00000001036      2327.00      1785.75 -0.38194109
```

##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 is used to indicate the organization of the index files as a matrix. We took the first column because that is the index and we used the unique() function to avoid obtaining any duplicate values.

How many genes do we have left?

```
nrow(mycounts)
```

```
[1] 21817
```

How many genes are "up" regulated at a threshold log2-fold-change of +2 or greater?

```
sum(mycounts$log2fc >= 2)
```

```
[1] 314
```

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

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

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

```
[1] 250
```

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

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

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

```
[1] 367
```

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

No, I don't trust these results because the output data is only obtained from taking the log of 2 and -2 but not actually providing any statistical data such as the p-value.

Time to do things the rest of the world do them with DESeq2

```
library(DESeq2)
```

```
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
```

```
#results(dds)
```

```
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     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
                     padj
                <numeric>
ENSG00000000003  0.163035
ENSG00000000005        NA
ENSG00000000419  0.176032
ENSG00000000457  0.961694
ENSG00000000460  0.815849
ENSG00000000938        NA
```

```r
#as.data.frame(res)
#View(as.data.frame(res))
```

```r
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
```

```r
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
```

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

```
Attaching package: 'AnnotationDbi'

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

    select
```

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

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

```
 [1] "ACCNUM"       "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"
```

```
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>
ENSG00000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
                     padj      symbol
                <numeric> <character>
ENSG00000000003  0.163035      TSPAN6
```

```
ENSG00000000005        NA        TNMD
ENSG00000000419  0.176032        DPM1
ENSG00000000457  0.961694       SCYL3
ENSG00000000460  0.815849    C1orf112
ENSG00000000938        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.

See codes below:

```
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>
ENSG00000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
                     padj      symbol      entrez     uniprot
                <numeric> <character> <character> <character>
ENSG00000000003  0.163035      TSPAN6        7105  A0A024RCI0
ENSG00000000005        NA        TNMD       64102      Q9H2S6
ENSG00000000419  0.176032        DPM1        8813      O60762
ENSG00000000457  0.961694       SCYL3       57147      Q8IZE3
ENSG00000000460  0.815849    C1orf112       55732  A0A024R922
ENSG00000000938        NA         FGR        2268      P09769
                           genename
                        <character>
ENSG00000000003          tetraspanin 6
ENSG00000000005            tenomodulin
ENSG00000000419 dolichyl-phosphate m..
ENSG00000000457 SCY1 like pseudokina..
ENSG00000000460 chromosome 1 open re..
ENSG00000000938 FGR proto-oncogene, ..
```

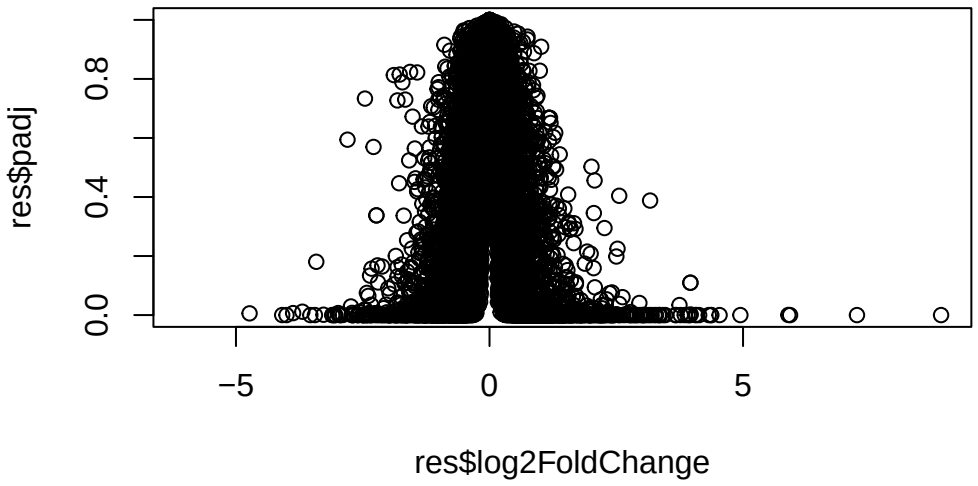
```r
ord <- order( res$padj )
#View(res[ord,])
head(res[ord,])
```

```
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
```

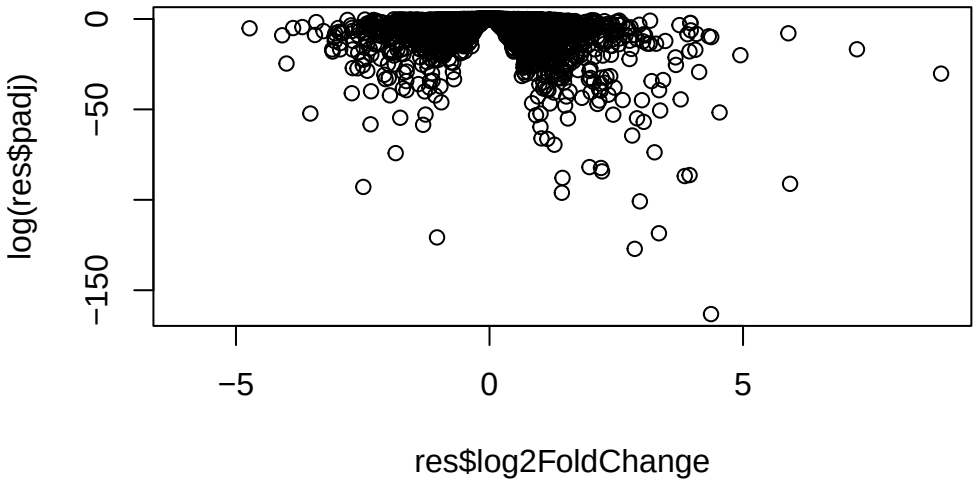
```
ENSG00000120129  3440.704        2.96521 0.2036951   14.5571 5.26424e-48
ENSG00000148175 13493.920        1.42717 0.1003890   14.2164 7.25128e-46
                      padj      symbol      entrez     uniprot
                 <numeric> <character> <character> <character>
ENSG00000152583 1.32441e-71     SPARCL1        8404  A0A024RDE1
ENSG00000179094 6.13966e-56        PER1        5187      O15534
ENSG00000116584 3.49776e-53     ARHGEF2        9181      Q92974
ENSG00000189221 3.46227e-52        MAOA        4128      P21397
ENSG00000120129 1.59454e-44       DUSP1        1843      B4DU40
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
```

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

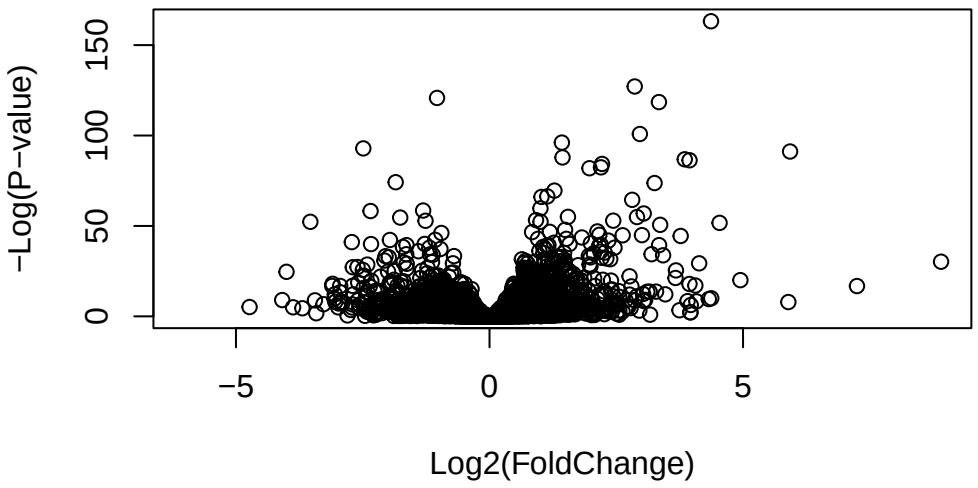
```
plot(res$log2FoldChange, res$padj)
```



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

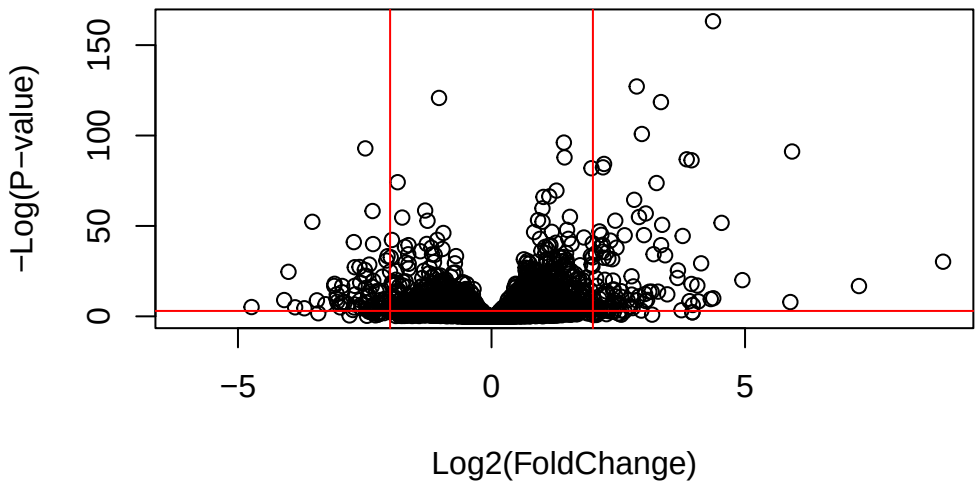



```
plot( res$log2FoldChange,  -log(res$padj),
      xlab="Log2(FoldChange)",
      ylab="-Log(P-value)")
```




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

# Add some cut-off lines
abline(v=c(-2,2), col="red", lty=1)
abline(h=-log(0.05), col="red", lty=1)
```

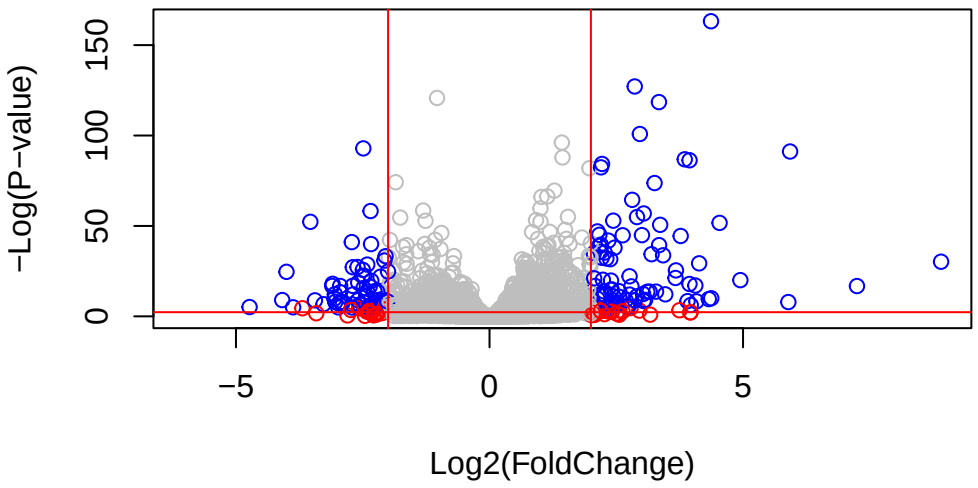



```
# Setup our custom point color vector
mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ]  <- "red"

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="red", lty=1)
abline(h=-log(0.1), col="red", lty=1)
```



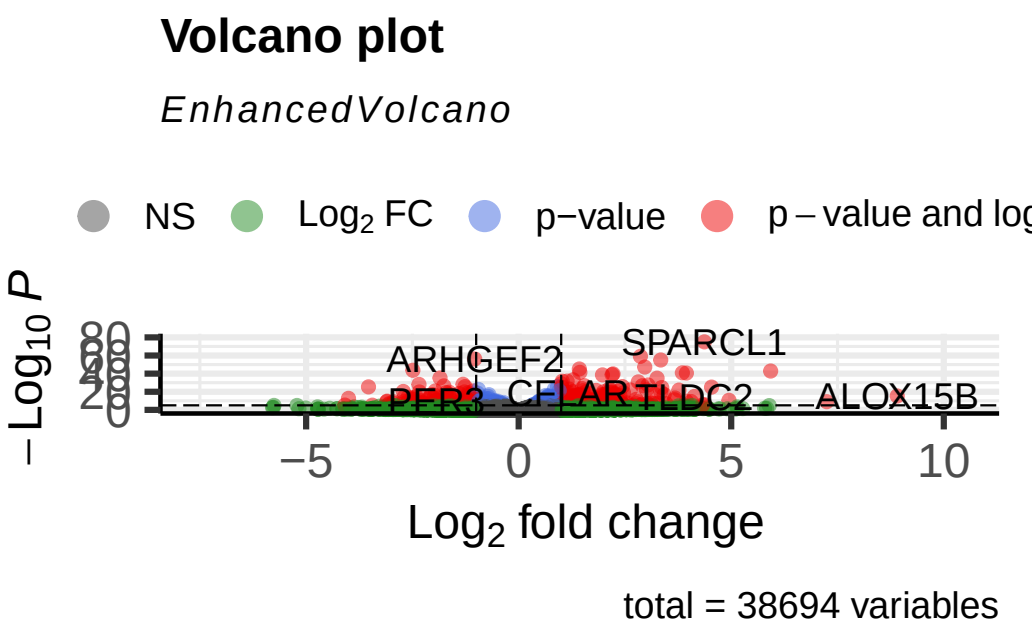

```
library(EnhancedVolcano)
```

```
Loading required package: ggrepel
```

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

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

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




```
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"
```

```
foldchanges = res$log2FoldChange
names(foldchanges) = res$entrez
head(foldchanges)
```

```
      7105       64102        8813       57147       55732        2268
-0.35070302          NA  0.20610777  0.02452695 -0.14714205 -1.73228897
```

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

```
attributes(keggres)
```

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

```
# 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
```

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

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

Info: Working in directory /Users/karenguerrero/Documents/BGGN 213/Week 7/class11

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

```
# A different PDF based output of the same data
pathview(gene.data=foldchanges, pathway.id="hsa05310", kegg.native=FALSE)
```

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

Info: Working in directory /Users/karenguerrero/Documents/BGGN 213/Week 7/class11

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

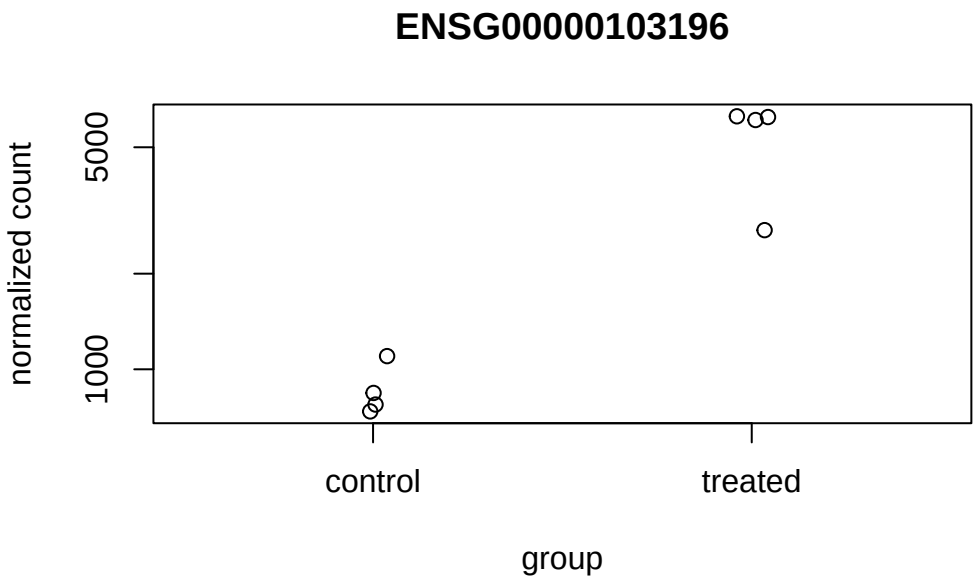
```
i <- grep("CRISPLD2", res$symbol)
res[i,]
```

```
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  A0A140VK80
                           genename
                        <character>
ENSG00000103196 cysteine rich secret..
```

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

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

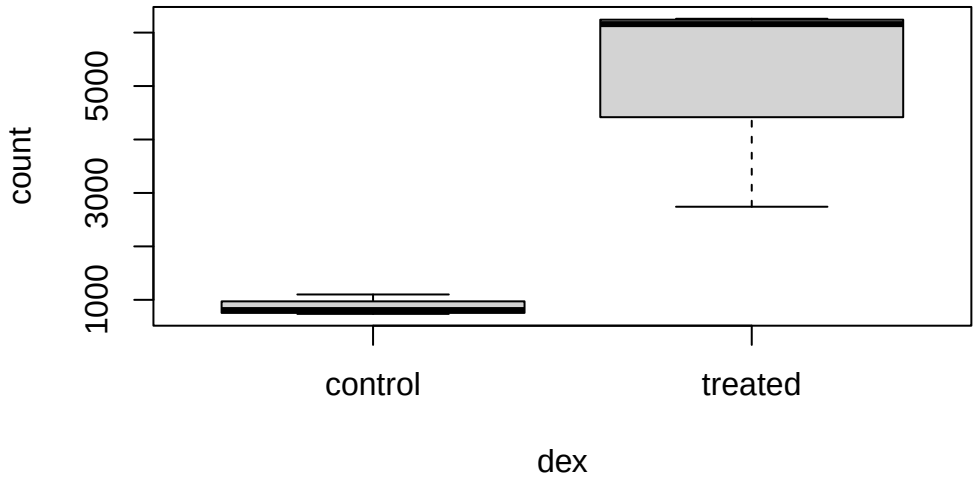
```
library(DESeq2)
plotCounts(dds, gene="ENSG00000103196", intgroup="dex")
```



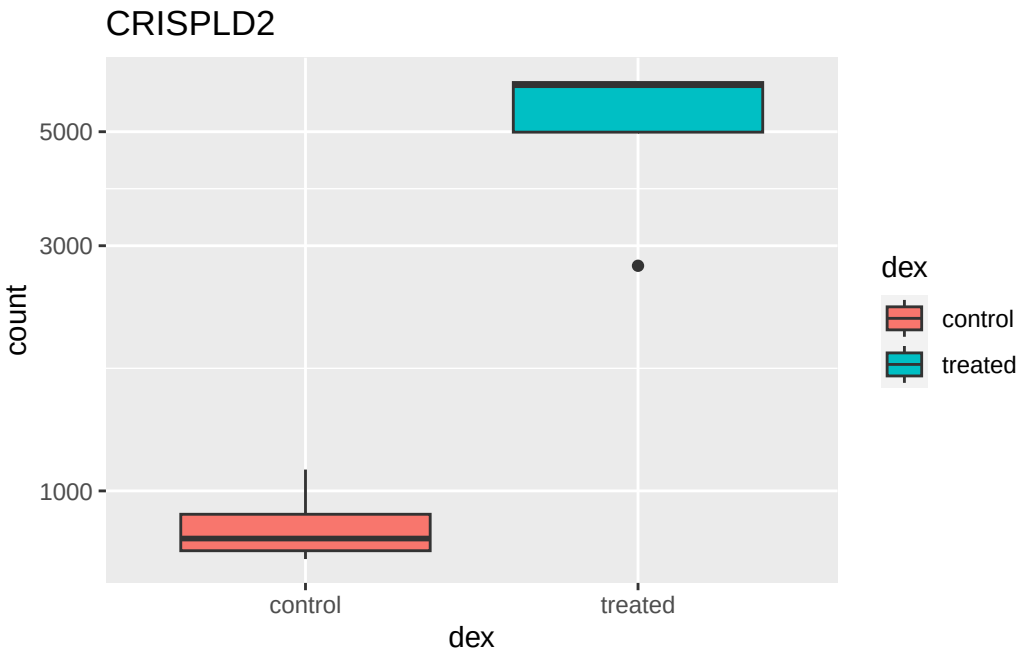

```
# Return the data
d <- plotCounts(dds, gene="ENSG00000103196", intgroup="dex", returnData=TRUE)
head(d)
```

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

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




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


CRISPLD2
5000
3000
1000
count
control
treated
dex
dex
control
treated