Class 13

Ashley Allen (PID: A14633373)

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

Import countData and colData

There are two data sets I need to import/read: - `countData` the transcript counts per gene (rows) in the different experiments - `colData` information (a.k.a. metadata) about the columns (i.e. experiments) in `countData`

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

Looking at the data with `head()`

```
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
```

```
head(metadata)
```

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

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

Q2. How many 'control' cell lines do we have?

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

```
control treated
      4       4
```

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

```
[1] 4
```

Toy differential gene expression

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

Step 1: Extract all "control" columns from the `counts` data

```
control.inds <- metadata$dex == "control"
control.counts <- counts[ , control.inds]
```

```
dim(control.counts)
```

```
[1] 38694     4
```

Step 2: Find the mean value for each gene in these columns

Now find the row wise mean

```
control.mean <- rowSums(control.counts)/ncol(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? Is there a function that could help here?

Rather than dividing by 4 we could divide by ncol() in order to make sure our mean is always correct.

> 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.inds <- metadata$dex == "treated"
treated.counts <- counts[ , treated.inds]
```

```
treated.mean <- rowSums(treated.counts)/ncol(treated.counts)
```

```
head(treated.mean)
```

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

Let's put these two mean values together

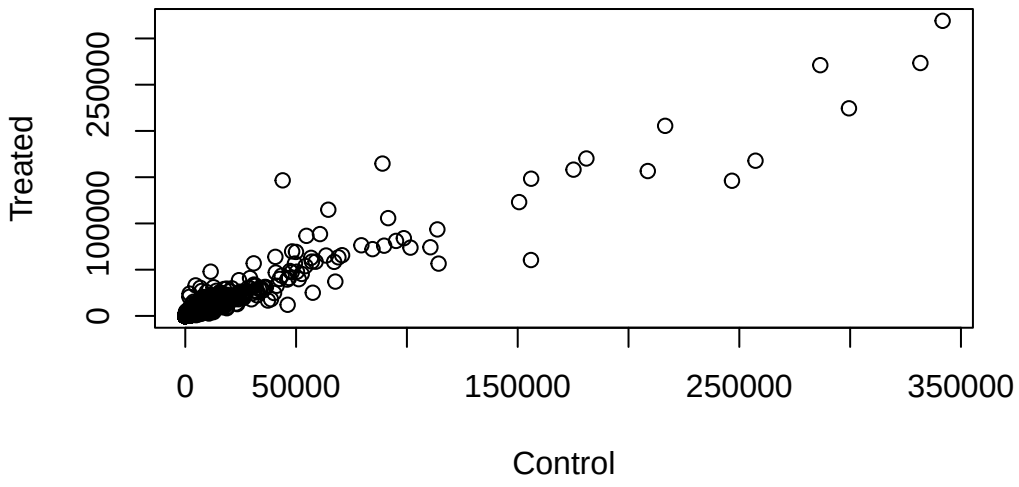
```
meancounts <- data.frame(control.mean, treated.mean)
```

```
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, xlab="Control", ylab="Treated")
```

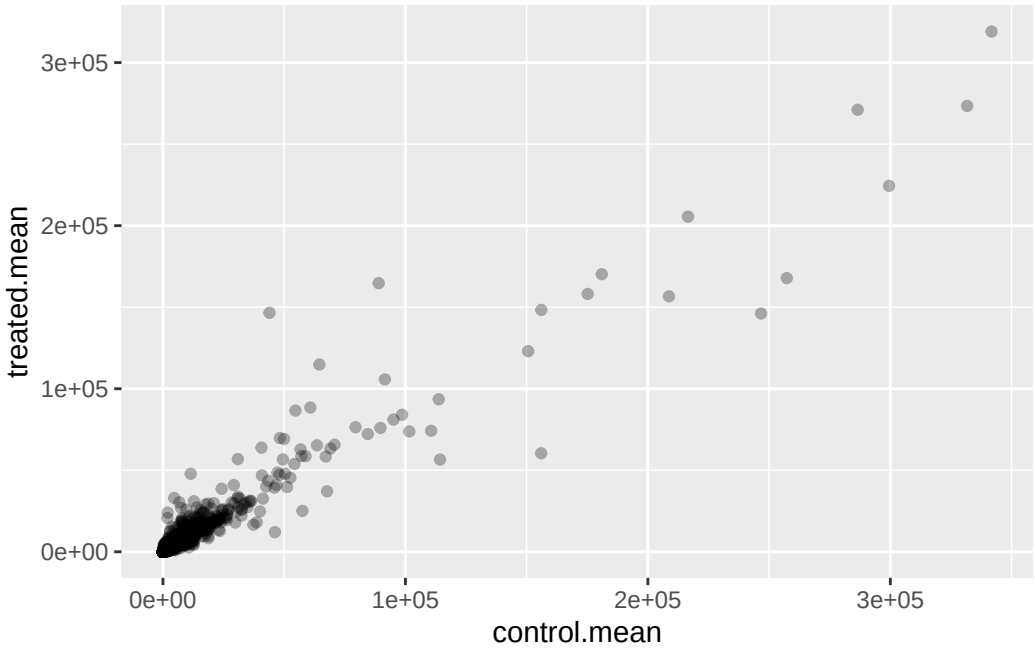


Q5 (b).You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot?

```
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 4.3.3
```

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



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

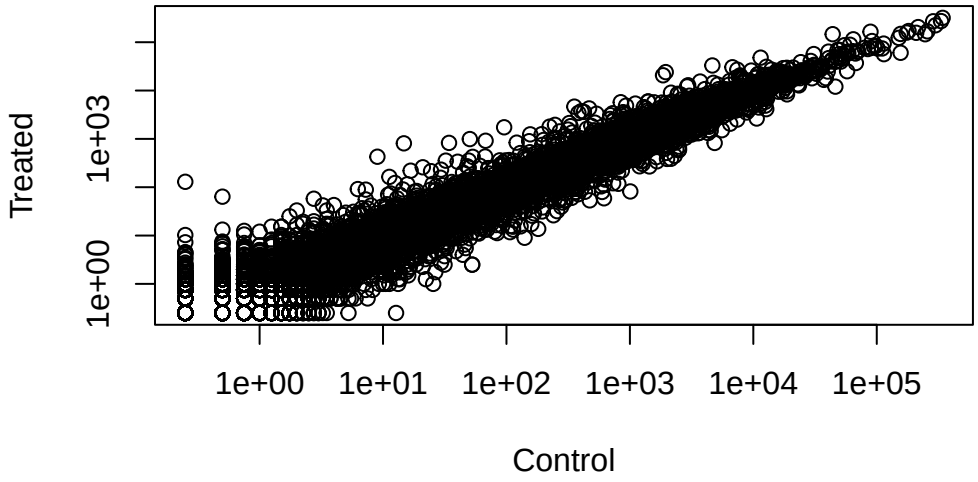
Adding `log="xy"` allows us to use a log scale using the `plot()` argument, but in `ggplot()` we can use `scale_x_continuous(trans="log2"` for x (and then chaning it for y as well).

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

```
plot(meancounts, xlab="Control", ylab="Treated", log="xy")
```

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

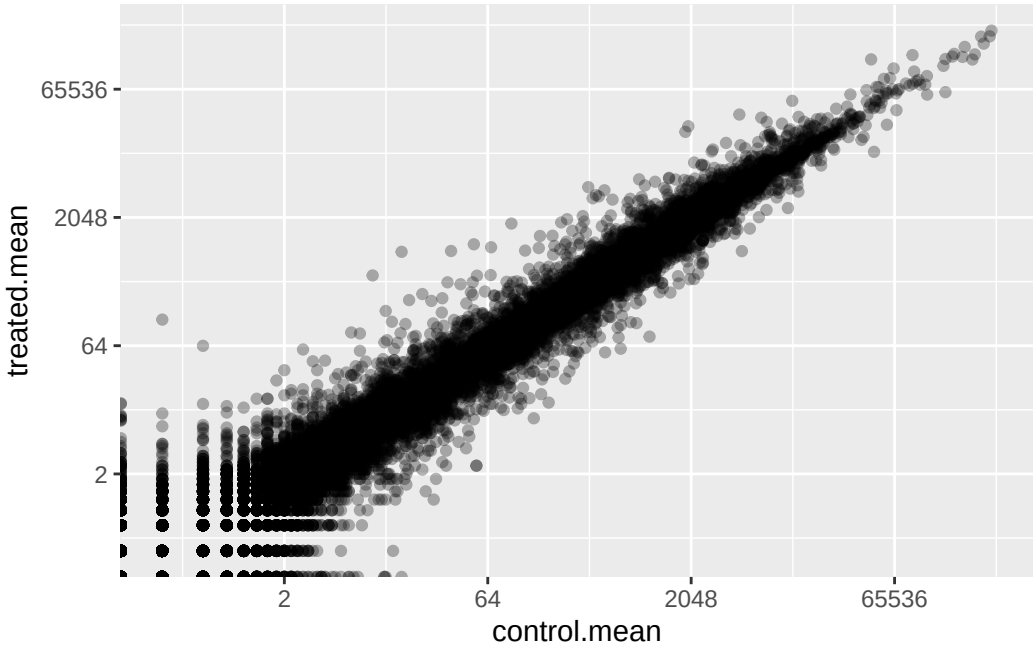
```
Warning in xy.coords(x, y, xlabel, ylabel, log): 15281 y values <= 0 omitted
from logarithmic plot
```




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



We most often work in log2 units as this makes the math easier.

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

```
[1] 0
```

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

```
[1] 1
```

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

```
[1] 2
```

```
# treated / control
log2(20/40)
```

```
[1] -1
```

Add "log2 fold-change" values to our `meancounts` data set

```
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
```

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

How many genes are "up" regulated at the common log2 fold-change threshold of +2?

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

```
[1] 1910
```

How many genes are "down" regulated at the threshold of -2?

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

```
[1] 2330
```

```
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
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.in` function is used to return TRUE values that will tell us which genes and samples have zero values. Unique would then ensure we don't count rows twice if they have both zero entries.

> Q8. Using the up.ind vector above can you determine how many up regulated genes we have at the 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? (unanswered)

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

```
[1] 367
```

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

I don't trust these results, as it just tallies TRUE and FALSE values.

Setting up for DESeq2

To do this the right way we need to consider the significance of the differences not just their magnitude.

```
#/ message:  false
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


Attaching package: 'IRanges'
```

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

    windows

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

Warning: package 'matrixStats' was built under R version 4.3.3


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
```

To use this package it wants countData and colData in a specific format.

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

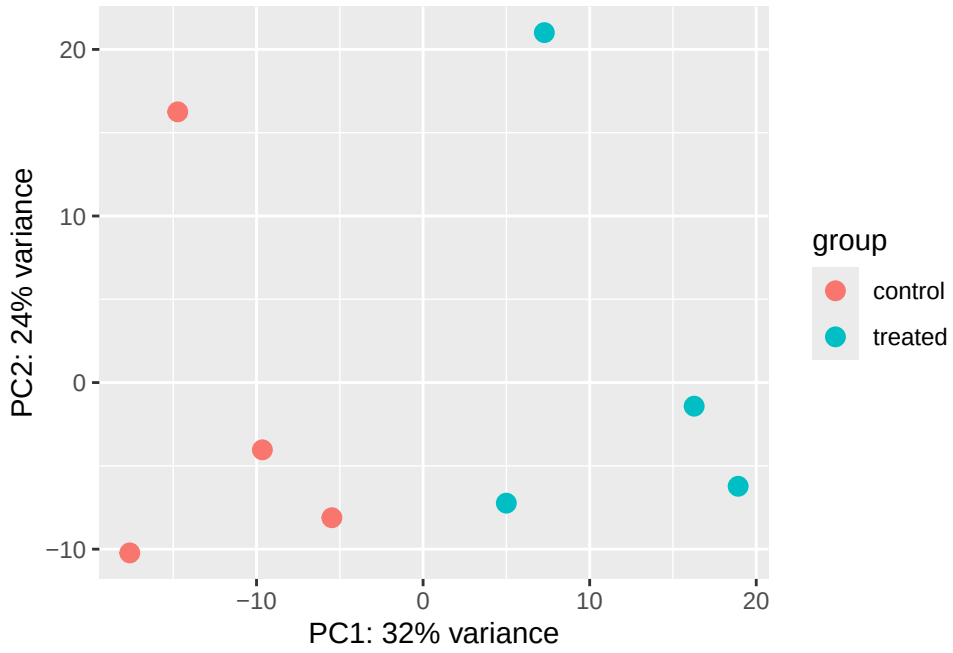
```
converting counts to integer mode

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

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
```

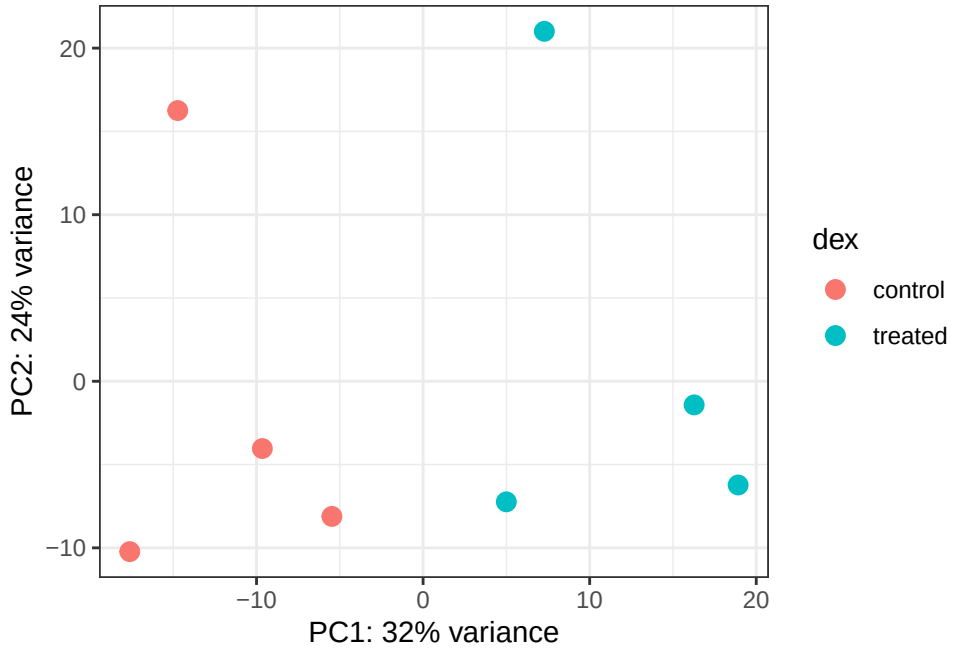
```
head(pcaData)
```

```
                 PC1        PC2   group     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()
```



DESeq analysis

```
dds <- DESeq(dds)
```

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

Extract my results:

```
res <- results(dds)
head(res)
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 6 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
                 <numeric>      <numeric> <numeric> <numeric> <numeric>
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
```

```
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
```

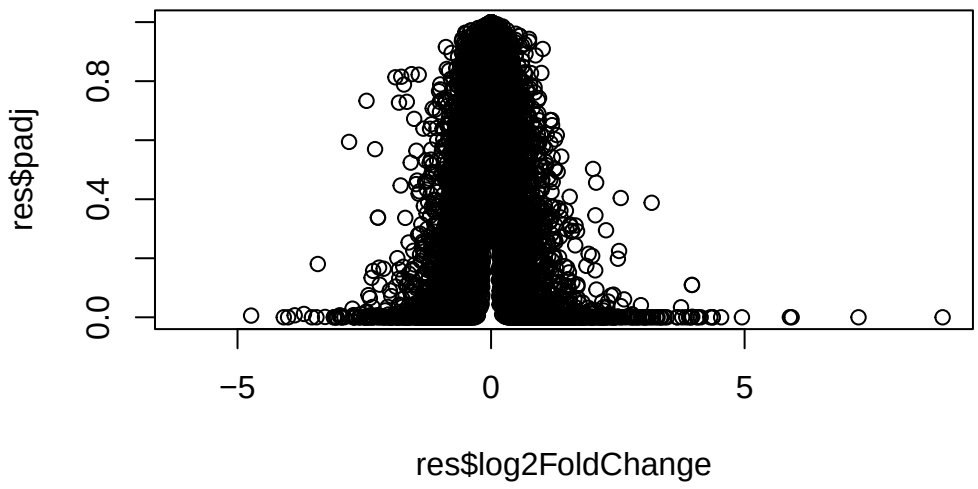
```
res05 <- results(dds, alpha=0.05)
summary(res05)
```

```
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
```

Data Visualization

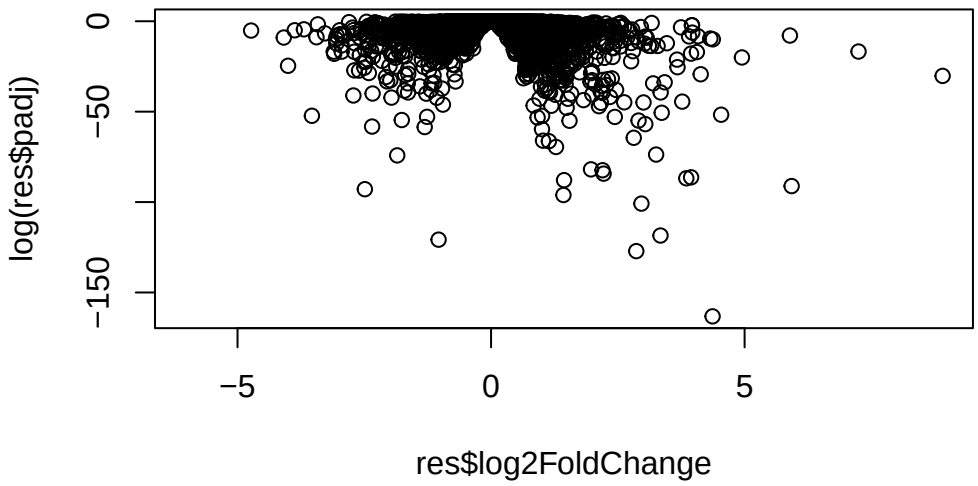
Plot of fold-change vs. P-value (adjusted for multiple testing)

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



Take the log of the P-value. Looking down the y-axis we see more significance.

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




```
log(0.01)
```

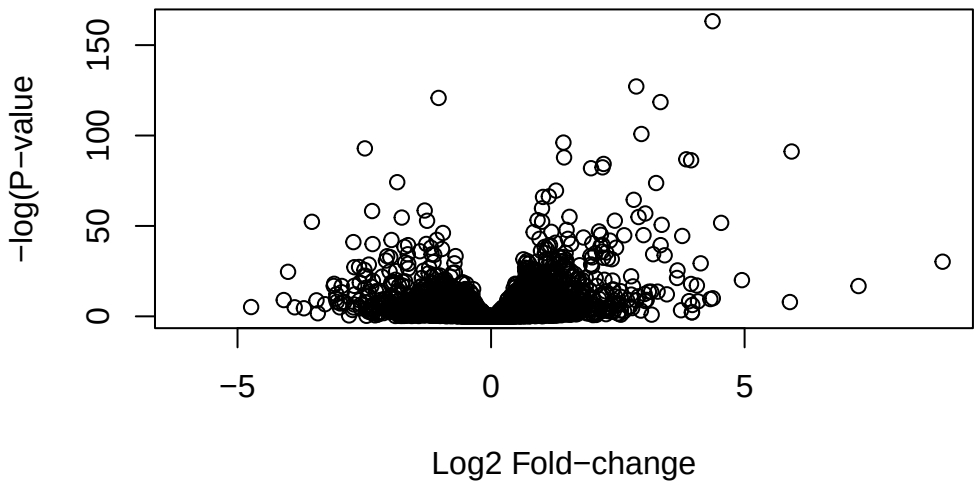
```
[1] -4.60517
```

```
log(0.0000000001)
```

```
[1] -23.02585
```

We can flip the y-axis (so we can read UP rather than DOWN the axis) to make it easier to read, by putting a minus sign on it (the result is a volcano plot).

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



Let's save our work to date

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

To finish off let's make a nicer volcano plot

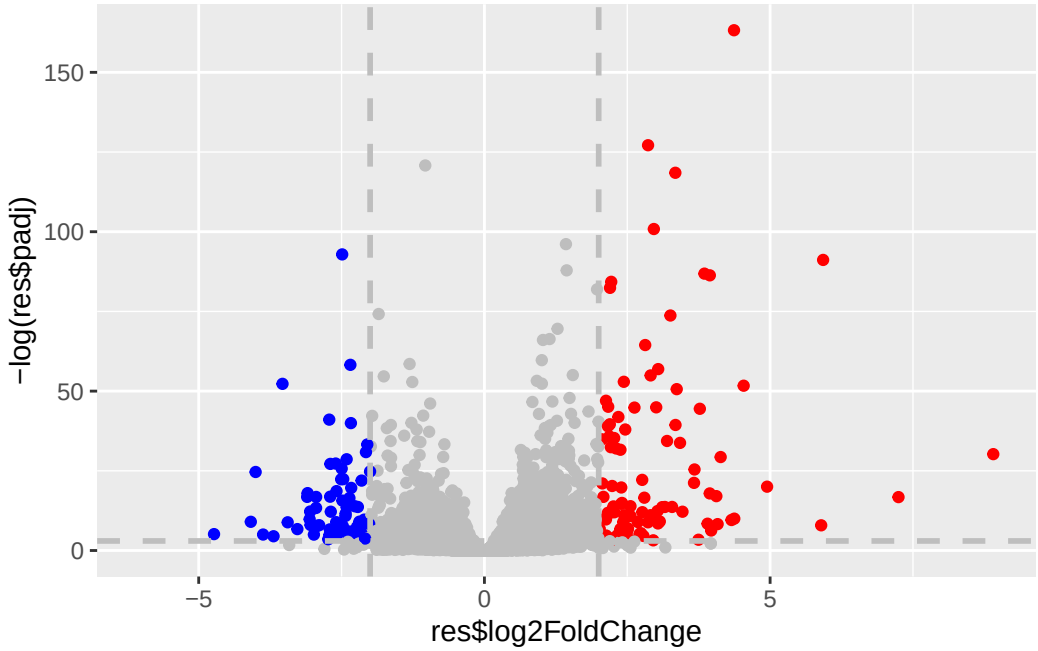
Step 1: Add the log2 threshold lines of +2/-2 **Step 2:** Add P-value threshold lines at 0.05 **Step 3:** Add color to highlight the subset of genes that meet both the above thresholds

```
mycols <- rep("grey", nrow(res))
mycols[res$log2FoldChange >=2] <- "red"
mycols[res$log2FoldChange <=-2] <- "blue"
mycols[res$padj > 0.05] <- "grey"
```

```
ggplot(res) +
  aes(res$log2FoldChange, -log(res$padj)) +
  geom_point(col=mycols) +
  geom_vline(xintercept=c(-2,2), linetype="dashed", color="grey", size=1) +
  geom_hline(yintercept=-log(0.05), linetype="dashed", color="grey", size=1)
```

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

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




```
#library(EnhancedVolcano)
#library(ggrepel)
```

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

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

Add gene annotation data

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

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

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

```
library(AnnotationDbi)
```

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

What database identifiers can I translate between here:

```
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"
```

We can use the `mapIDS()` function to translate/map between these different identifier formats.

> 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.

Let's add SYMBOL, GENENAME, and ENTREZID

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

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

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

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

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

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

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

```
'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               genename      entrez
                 <numeric> <character>            <character> <character>
ENSG00000000003  0.163035      TSPAN6          tetraspanin 6        7105
ENSG00000000005        NA        TNMD            tenomodulin       64102
ENSG00000000419  0.176032        DPM1 dolichyl-phosphate m..        8813
ENSG00000000457  0.961694       SCYL3 SCY1 like pseudokina..       57147
ENSG00000000460  0.815849       FIRRM FIGNL1 interacting r..       55732
ENSG00000000938        NA         FGR FGR proto-oncogene, ..        2268
                    uniprot
                 <character>
ENSG00000000003  A0A024RCI0
ENSG00000000005      Q9H2S6
ENSG00000000419      O60762
```

```
ENSG00000000457      Q8IZE3
ENSG00000000460  A0A024R922
ENSG00000000938      P09769
```

```
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               genename      entrez
                 <numeric> <character>            <character> <character>
ENSG00000152583 1.32441e-71     SPARCL1           SPARC like 1        8404
ENSG00000179094 6.13966e-56        PER1 period circadian reg..        5187
ENSG00000116584 3.49776e-53     ARHGEF2 Rho/Rac guanine nucl..        9181
ENSG00000189221 3.46227e-52        MAOA    monoamine oxidase A        4128
ENSG00000120129 1.59454e-44       DUSP1 dual specificity pho..        1843
ENSG00000148175 1.83034e-42        STOM               stomatin        2040
                    uniprot
                <character>
ENSG00000152583  A0A024RDE1
ENSG00000179094      O15534
ENSG00000116584      Q92974
ENSG00000189221      P21397
ENSG00000120129      B4DU40
ENSG00000148175      F8VSL7
```

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

Pathway analysis

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

This is the job of "pathway analysis" (a.k.a. "gene set enrichment")

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

```
library(gage)
```

```
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(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"
```

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

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

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

We can add names to our vector that are useful for book keeping so we know what a given value corresponds to:

```
x <- c(10,100,200)
names(x) <- c("barry", "alice", "chandra")
x
```

```
  barry   alice chandra
     10     100     200
```

Let's put names on our `foldchanges` vector - here we will use `res$entrez`

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

Now I can run "pathway analysis"

```
keggres = gage(foldchanges, gsets=kegg.sets.hs)
attributes(keggres)
```

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

```
head(keggres$less)
```

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

```
hsa04940 Type I diabetes mellitus                      0.0017820293 0.14232581
hsa05310 Asthma                                        0.0020045888 0.14232581
hsa04672 Intestinal immune network for IgA production 0.0060434515 0.31387180
hsa05330 Allograft rejection                           0.0073678825 0.31387180
hsa04340 Hedgehog signaling pathway                    0.0133239547 0.47300039
                                                       set.size         exp1
hsa05332 Graft-versus-host disease                           40 0.0004250461
hsa04940 Type I diabetes mellitus                            42 0.0017820293
hsa05310 Asthma                                              29 0.0020045888
hsa04672 Intestinal immune network for IgA production        47 0.0060434515
hsa05330 Allograft rejection                                 36 0.0073678825
hsa04340 Hedgehog signaling pathway                          56 0.0133239547
```

```
# 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
```

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

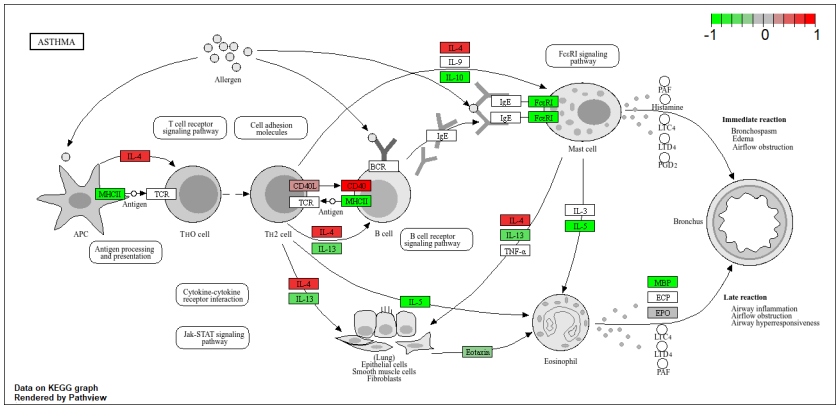
```
pathview(foldchanges, pathway.id = "hsa05310")
```

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

```
Info: Working in directory C:/Users/allen/OneDrive/Desktop/BIMM 143/class/class13
```

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

Insert this figure in my report



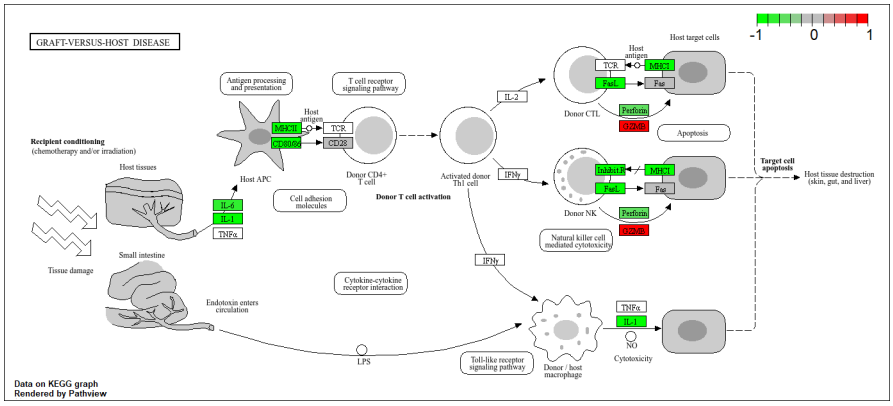
Q12. Can you do the same procedure as above to plot the pathview figures for the top 2 down-reguled pathways?

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

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

```
Info: Working in directory C:/Users/allen/OneDrive/Desktop/BIMM 143/class/class13
```

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

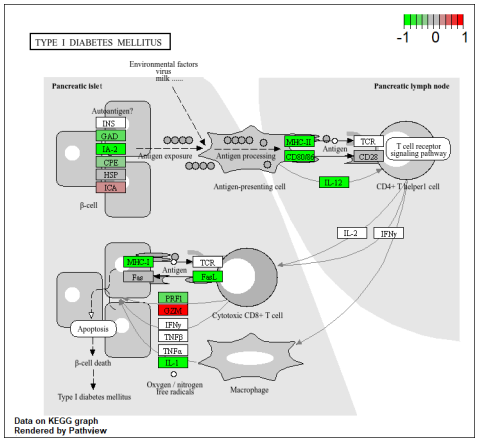



```
pathview(foldchanges, pathway.id = "hsa04940")
```

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

```
Info: Working in directory C:/Users/allen/OneDrive/Desktop/BIMM 143/class/class13
```

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



Plotting Counts for Genes of Interest

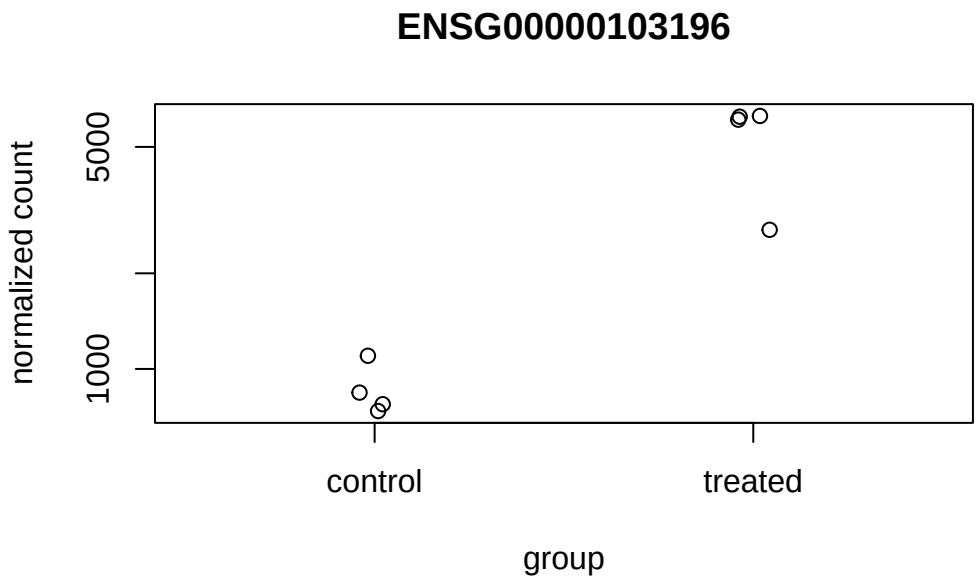
```
i <- grep("CRISPLD2", res$symbol)
res[i,]
```

```
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               genename      entrez
                 <numeric> <character>            <character> <character>
ENSG00000103196 3.36344e-20    CRISPLD2 cysteine rich secret..       83716
                   uniprot
                <character>
ENSG00000103196  A0A140VK80
```

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

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

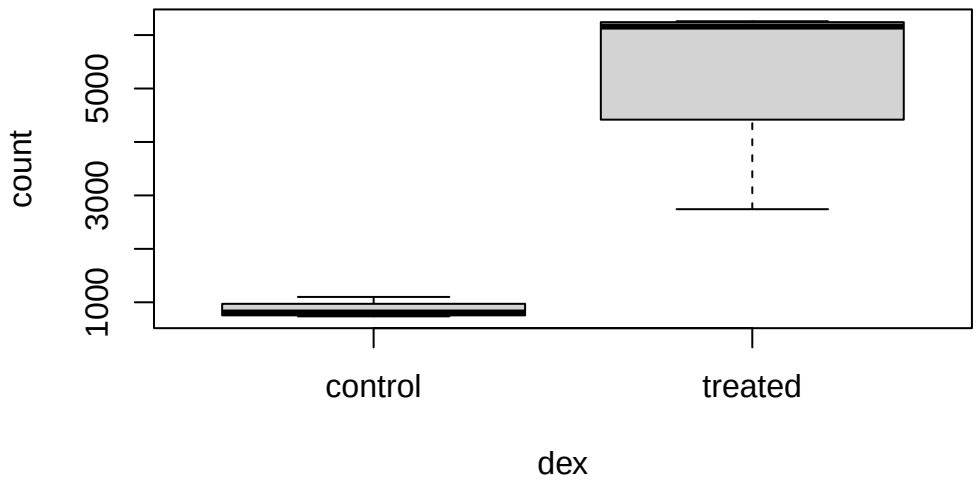
```
plotCounts(dds, gene="ENSG00000103196", intgroup="dex")
```



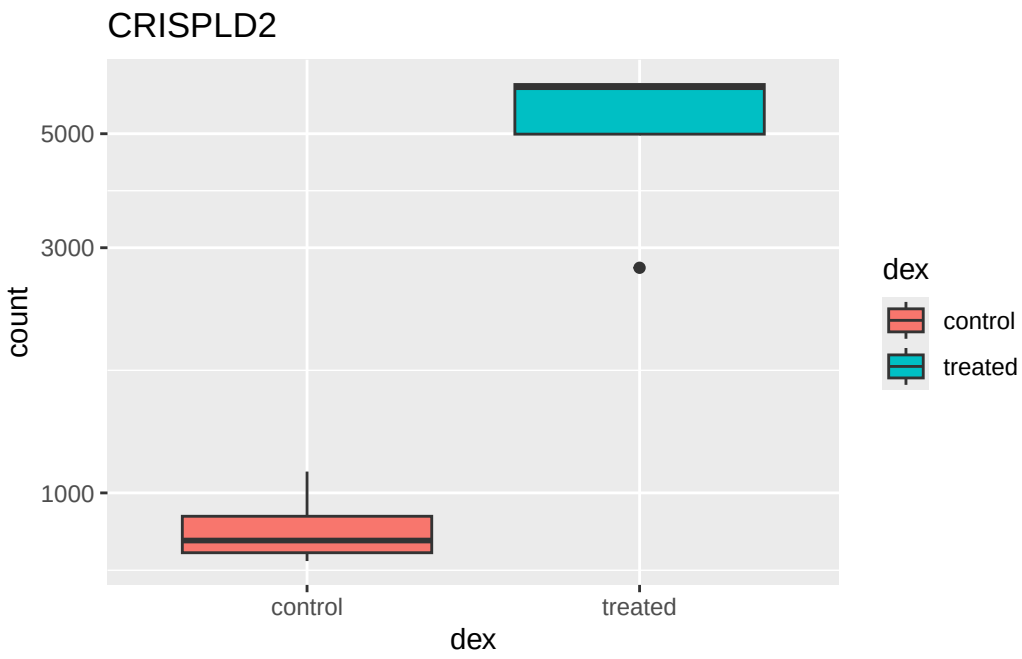

```
# Return the data
d <- plotCounts(dds, gene="ENSG00000103196", intgroup="dex", returnData=TRUE)
head(d)
```

```
               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