

Class 15 RNASeq Analysis

Groot (PID: A15485151)

11/16/2021

#BAckground Our data for today comes from Himes et al. RNASeq analysis of the drug dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al. 2014).

Read the countData and colData

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

Let's have a look at these

```
metadata
```

```
##          id      dex celltype     geo_id
## 1 SRR1039508 control    N61311 GSM1275862
## 2 SRR1039509 treated    N61311 GSM1275863
## 3 SRR1039512 control    N052611 GSM1275866
## 4 SRR1039513 treated    N052611 GSM1275867
## 5 SRR1039516 control    N080611 GSM1275870
## 6 SRR1039517 treated    N080611 GSM1275871
## 7 SRR1039520 control    N061011 GSM1275874
## 8 SRR1039521 treated    N061011 GSM1275875
```

```
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
```

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
## [1] 38694
```

38694 genes

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

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

```
## [1] 4
```

4 control cell lines

#Toy differential gene expression First I need to extract all the “control” columns. Then I wil take the rowwise mean to get the average count values for all genes in these four experiments.

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

```
##          SRR1039508 SRR1039512 SRR1039516 SRR1039520  
## ENSG000000000003     723      904     1170      806  
## ENSG000000000005      0        0        0        0  
## ENSG000000000419     467      616      582      417  
## ENSG000000000457     347      364      318      330  
## ENSG000000000460      96       73      118      102  
## ENSG000000000938      0        1        2        0
```

```
control mean <- rowSums( control counts )/4
```

Q3. How would you make the above code in either approach more robust?

Use instead rowMeans instead to prepare when the sample size is greater than 4.

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

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$dex == "treated"  
treated counts <- counts[, treated inds]  
treated mean <- rowMeans(treated counts)  
names(treated mean) <- counts$ensgene
```

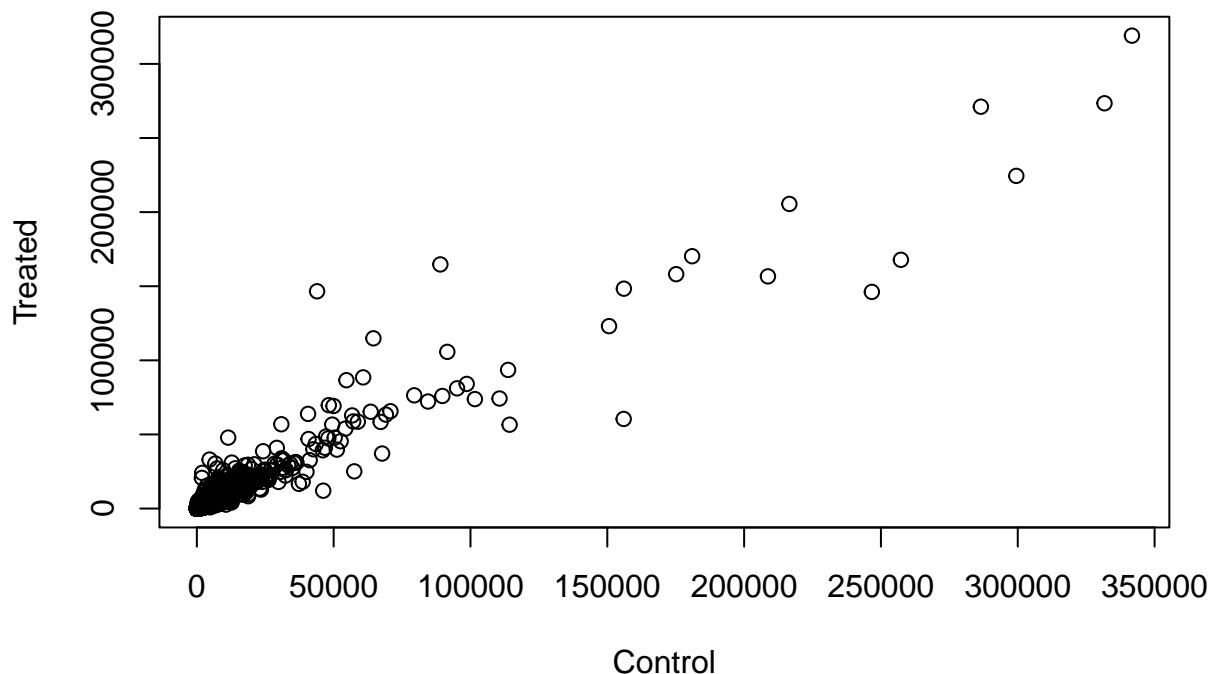
We will combine our meancount data for bookkeeping purposes

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

```
## control.mean treated.mean
##      23005324      22196524
```

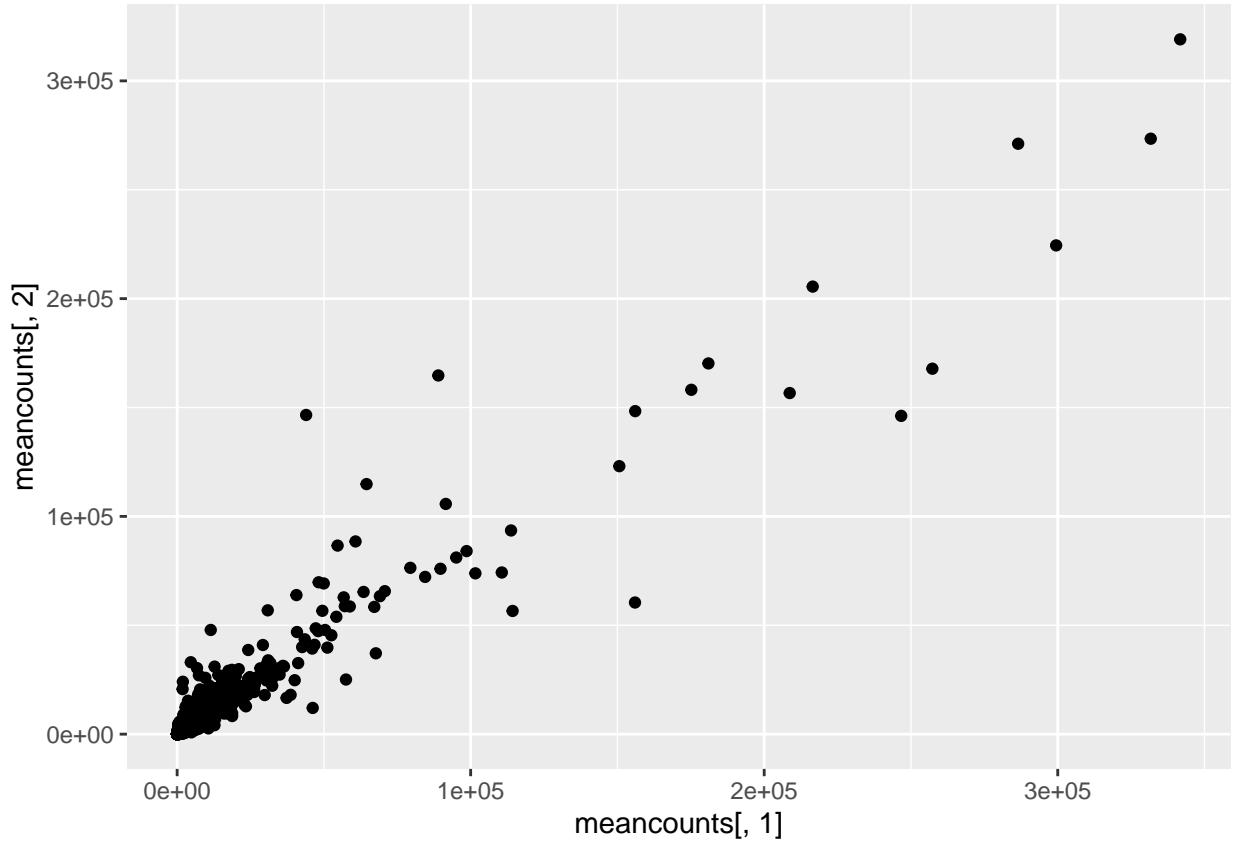
Q5 (a). Create a scatter plot showing the mean of the treated samples against the mean of the control samples. Your plot should look something like the following.

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



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

```
library(ggplot2)
ggplot(meancounts) + aes(x= meancounts[,1], y=meancounts[,2], xlab="Control", ylab="Treated") + geom_poo
```



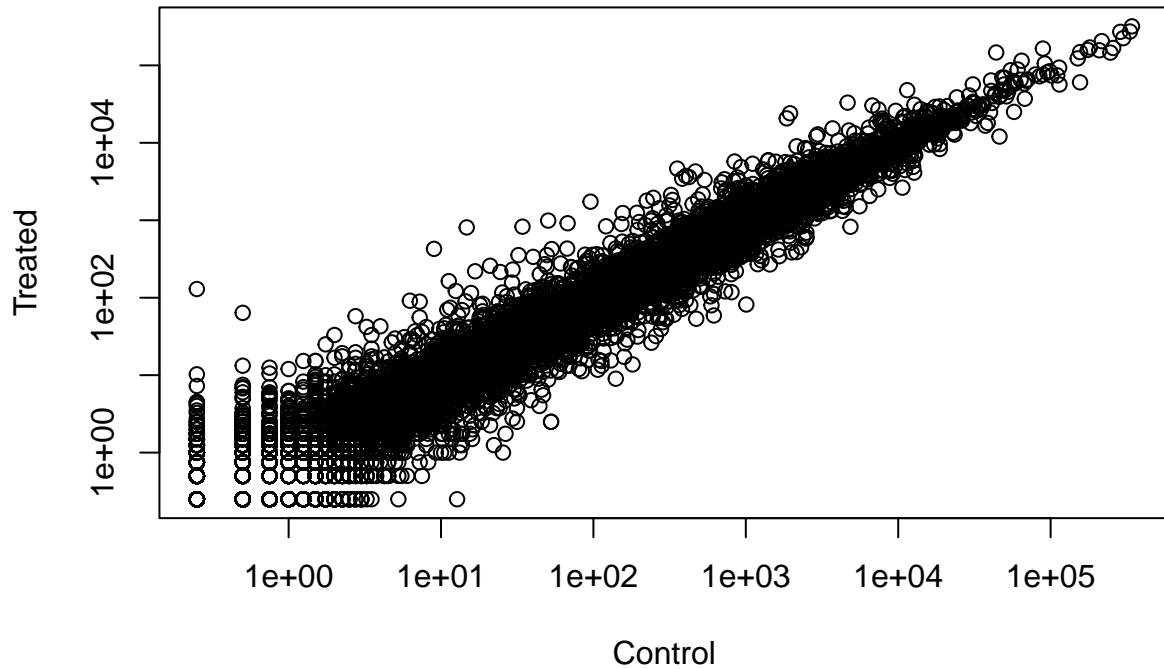
Q6. Try plotting both axes on a log scale. What is the argument to `plot()` that allows you to do this?

`log = "xy"`

```
plot(meancounts[,1], meancounts[,2], 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
```



We often use \log_2 in this field because it has nice math properties that make interpretation easier

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

```
## [1] 0
```

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

```
## [1] 2
```

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

```
## [1] -1
```

Cool we see 0 values for no change and + values for increases and minus values for decreases. This nice property leads us to work with **log2(fold-change)** all the time in the genomics and proteomics field.

```
meancounts$log2fc <- log2(meancounts[, "treated.mean"] / meancounts[, "control.mean"])
head(meancounts)
```

```
##                                     control.mean treated.mean      log2fc
## ENSG000000000003           900.75     658.00 -0.45303916
## ENSG000000000005            0.00      0.00        NaN
```

```

## ENSG00000000419      520.50      546.00  0.06900279
## ENSG00000000457      339.75      316.50 -0.10226805
## ENSG00000000460      97.25       78.75 -0.30441833
## ENSG00000000938      0.75        0.00    -Inf

```

I need to exclude the genes (i.e. rows) with zero counts because we can't say anything about these as we have no data for them!

```
which(c(F,F,T,T))
```

```
## [1] 3 4
```

I can use the **which()** function with the 'arr.ind=TRUE' argument to get the columns and rows where the TRUE values are (i.e. the zero counts in our case).

```

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

```

How many do we have left?

```
nrow(mycounts)
```

```
## [1] 21817
```

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

arr.ind = TRUE gives us both the columns and rows that have zero counts the unique() prevents rows contains more than one zeroes from being counted twice

Are the genes up-regulated 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?

```
table(up.ind) ["TRUE"]
```

```
## TRUE  
## 250
```

250 up-regulated genes

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

```
table(down.ind) ["TRUE"]
```

```
## TRUE  
## 367
```

367 down-regulated genes

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

No, because I have no information about the p-values

```
##DESeq2 analysis
```

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.

```
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, 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 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

```

We need to first set up the input object for the 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

```

Now we can run DESeq2 analysis

```

dds <- DESeq(dds)

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

```

To get at the results here we use the deseq ‘results()’ function

```

res <- results(dds)
res

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

```

convert the previous result into a dataframe

```

data.frame <- as.data.frame(res)
View(data.frame)

```

Summarize the data

```

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

```

```
reset p-value to 0.05
```

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

```
##  
## out of 25258 with nonzero total read count  
## adjusted p-value < 0.05  
## LFC > 0 (up) : 1236, 4.9%  
## LFC < 0 (down) : 933, 3.7%  
## outliers [1] : 142, 0.56%  
## low counts [2] : 9033, 36%  
## (mean count < 6)  
## [1] see 'cooksCutoff' argument of ?results  
## [2] see 'independentFiltering' argument of ?results
```

```
#Adding annotation data
```

```
#BiocManager::install("AnnotationDbi")
library("AnnotationDbi")
```

```
## Warning: package 'AnnotationDbi' was built under R version 4.1.2
```

```
#BiocManager::install("org.Hs.eg.db")
library("org.Hs.eg.db")
```

```
##
```

What are the key types?

```
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"
```

Add individual columns to result table using mapIds()

```
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.

```

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
## ENSG000000000419 520.134160  0.2061078  0.101059  2.039475 0.0414026
## ENSG000000000457 322.664844  0.0245269  0.145145  0.168982 0.8658106
## ENSG000000000460 87.682625 -0.1471420  0.257007 -0.572521 0.5669691
## ENSG000000000938 0.319167 -1.7322890  3.493601 -0.495846 0.6200029
##          padj      symbol      entrez      uniprot
##          <numeric> <character> <character> <character>
## ENSG000000000003 0.163035    TSPAN6      7105 AOA024RCIO
## ENSG000000000005      NA      TNMD      64102 Q9H2S6
## ENSG000000000419 0.176032    DPM1      8813 060762
## ENSG000000000457 0.961694    SCYL3      57147 Q8IZE3
## ENSG000000000460 0.815849   C1orf112     55732 AOA024R922
## ENSG000000000938      NA      FGR      2268 P09769
##          genename
##          <character>
## ENSG000000000003      tetraspanin 6
## ENSG000000000005      tenomodulin
## ENSG000000000419      dolichyl-phosphate m..
## ENSG000000000457      SCY1 like pseudokina..
## ENSG000000000460      chromosome 1 open re..
## ENSG000000000938      FGR proto-oncogene, ..

```

arrange results by the adjusted p-value and view

```

ord <- order( res$padj )
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 AOA024RDE1
## ENSG00000179094 6.13966e-56      PER1      5187 015534
## 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

```

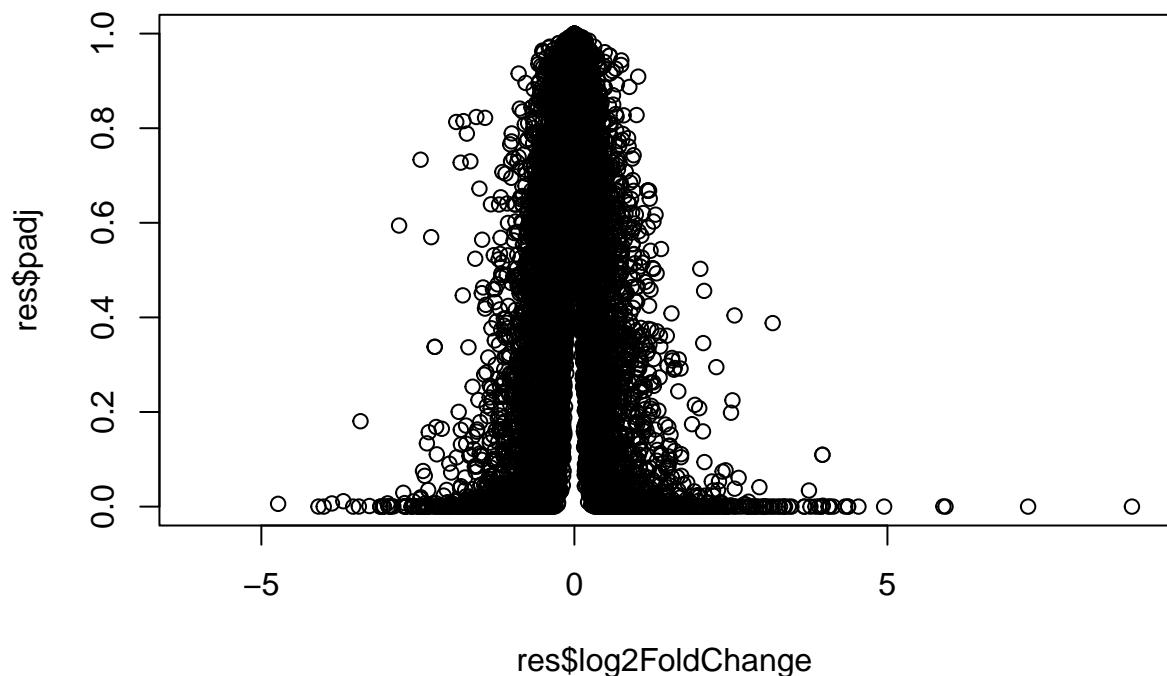
export results to .csv

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

##Data visualization

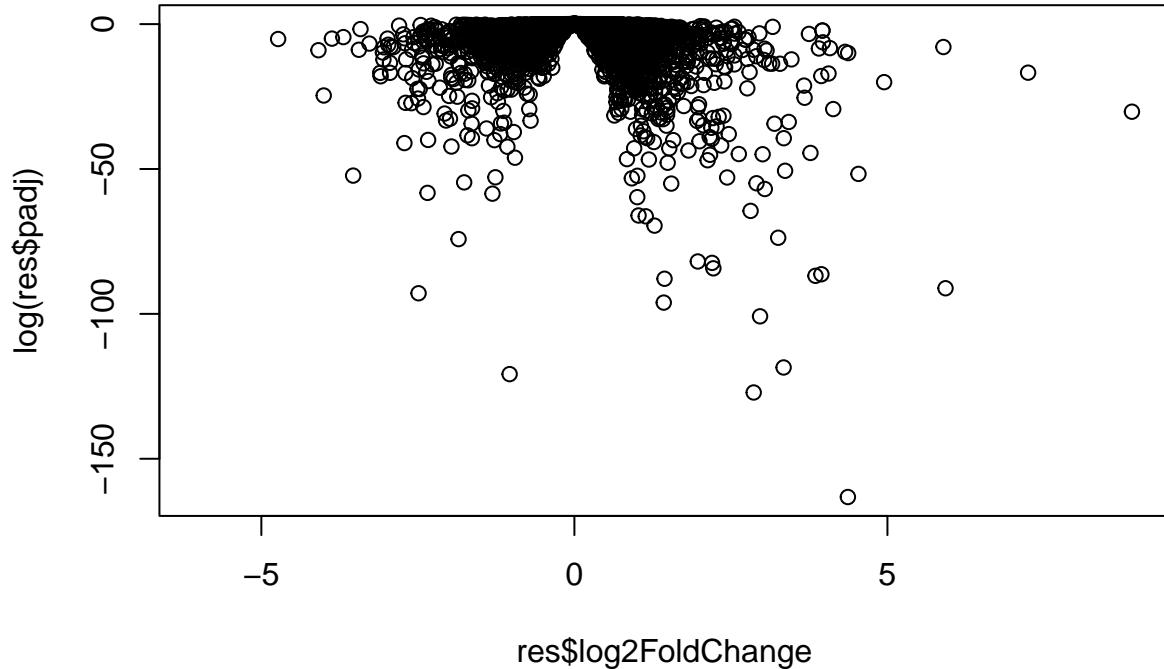
#volcano plot Let's make a commonly produced visualization from this data, namely a so-called Volcano plot. These summary figures are frequently used to highlight the proportion of genes that are both significantly regulated and display a high fold change.

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



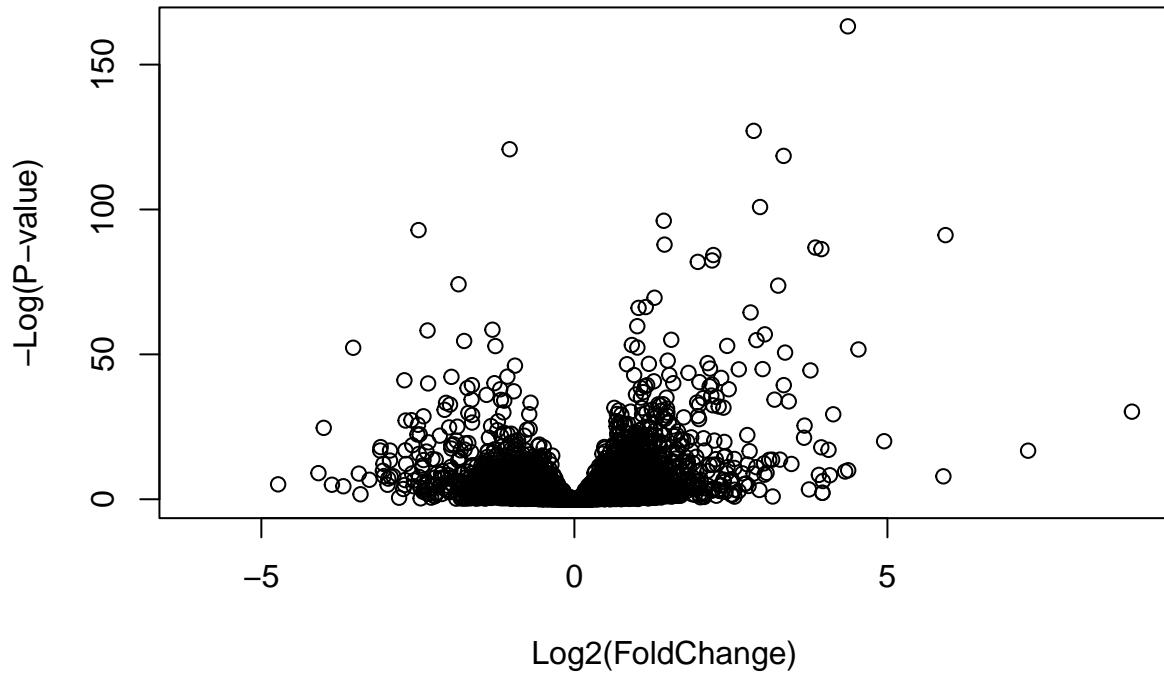
That is not a useful plot because all the small p-values are hidden at the bottom of the plot and we can't really see them. Log will help.

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



we can flip this pvalue axis by just adding a negative sign

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

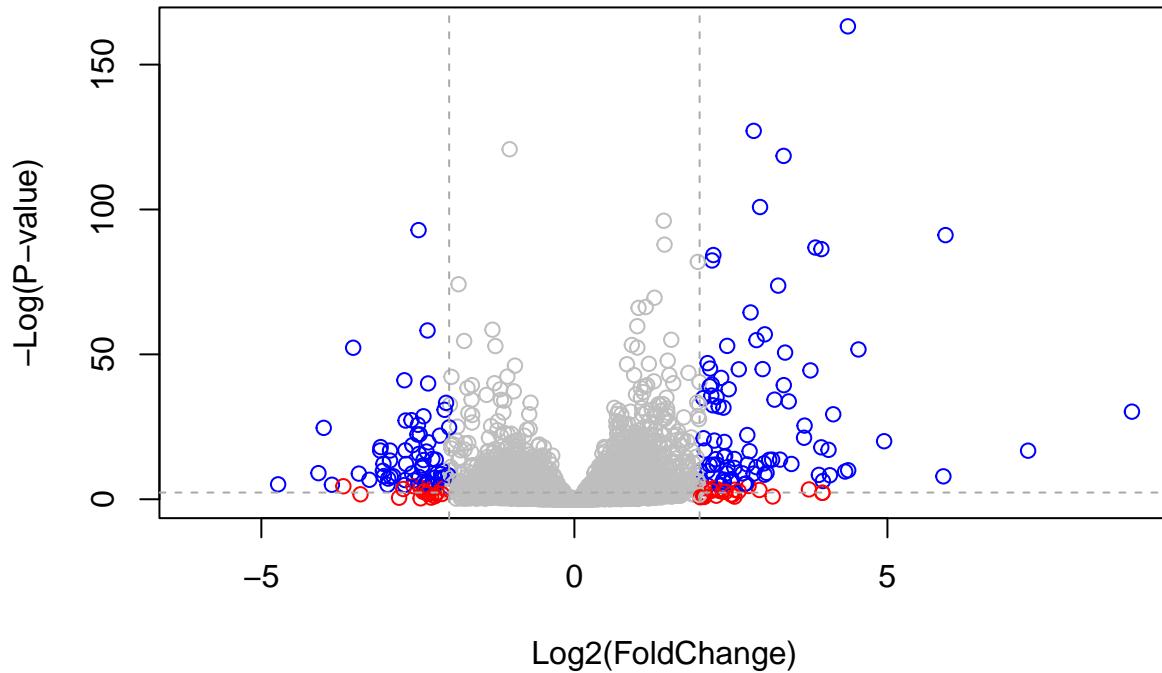


Finally let's add some color to this plot to draw attention to the genes (i.e. points) we care about - that is those with large fold-change and low pvalues (i.e. high -log(pvalues))

```
mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ] <- "red"
inds <- (res$padj < 0.01) & (abs(res$log2FoldChange) > 2 )
mycols[ inds ] <- "blue"
```

resulting 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="dark gray", lty=2)
abline(h=-log(0.1), col="dark gray", lty=2)
```



```
##Pathway analysis
```

```
# Run in your R console (i.e. not your Rmarkdown doc!)
#BiocManager::install( c("pathview", "gage", "gageData") )
```

```
set up KEGG dataset
```

```
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"
```

Prep before using 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
```

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
```

use pathview() function to make a pathway plot

```

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

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

## Info: Working in directory /Users/kevin/Desktop/BIMM 143/R Studio Project/bimm143_github/Class15

## 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/kevin/Desktop/BIMM 143/R Studio Project/bimm143_github/Class15

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

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

#Graft-versus-host disease
pathview(gene.data=foldchanges, pathway.id="hsa05332")

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

## Info: Working in directory /Users/kevin/Desktop/BIMM 143/R Studio Project/bimm143_github/Class15

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

pathview(gene.data=foldchanges, pathway.id="hsa05332", kegg.native=FALSE)

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

## Warning in .subtypeDisplay(object): Given subtype 'missing interaction' is not found!

## Info: Working in directory /Users/kevin/Desktop/BIMM 143/R Studio Project/bimm143_github/Class15

## Info: Writing image file hsa05332.pathview.pdf

#Type I diabetes mellitus
pathview(gene.data=foldchanges, pathway.id="hsa04940")

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

## Info: Working in directory /Users/kevin/Desktop/BIMM 143/R Studio Project/bimm143_github/Class15

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

```

```

pathview(gene.data=foldchanges, pathway.id="hsa04940", kegg.native=FALSE)

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

## Info: Working in directory /Users/kevin/Desktop/BIMM 143/R Studio Project/bimm143_github/Class15

## Info: Writing image file hsa04940.pathview.pdf

##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..

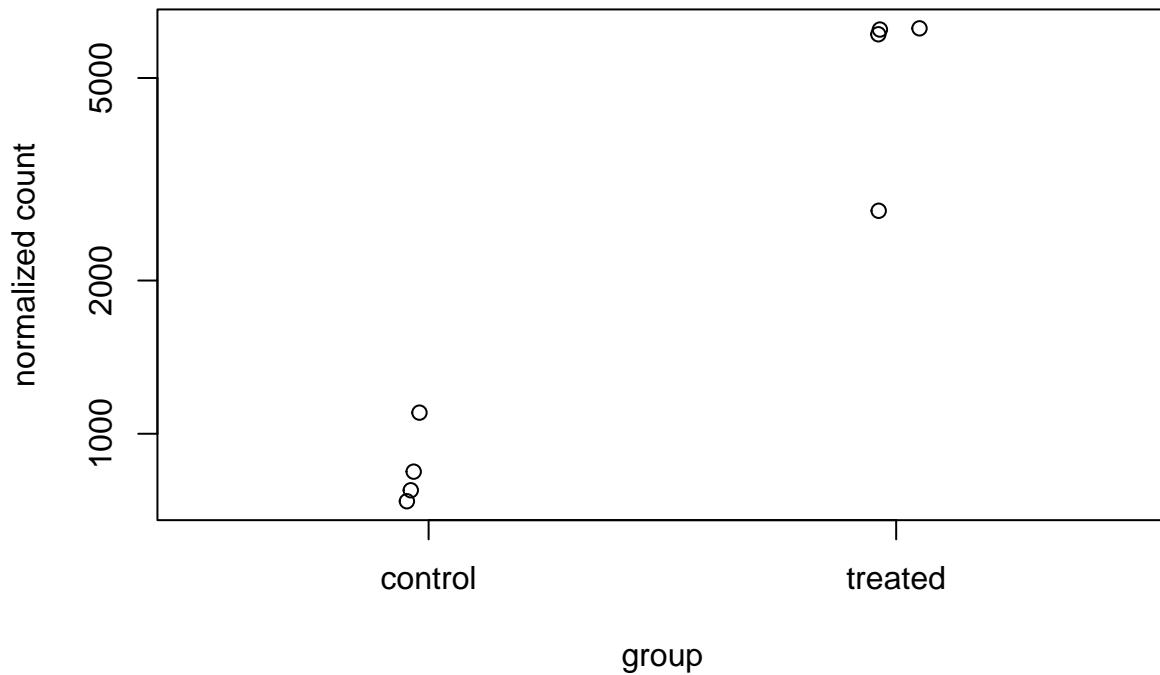
rownames(res[i,])

## [1] "ENSG00000103196"

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

```

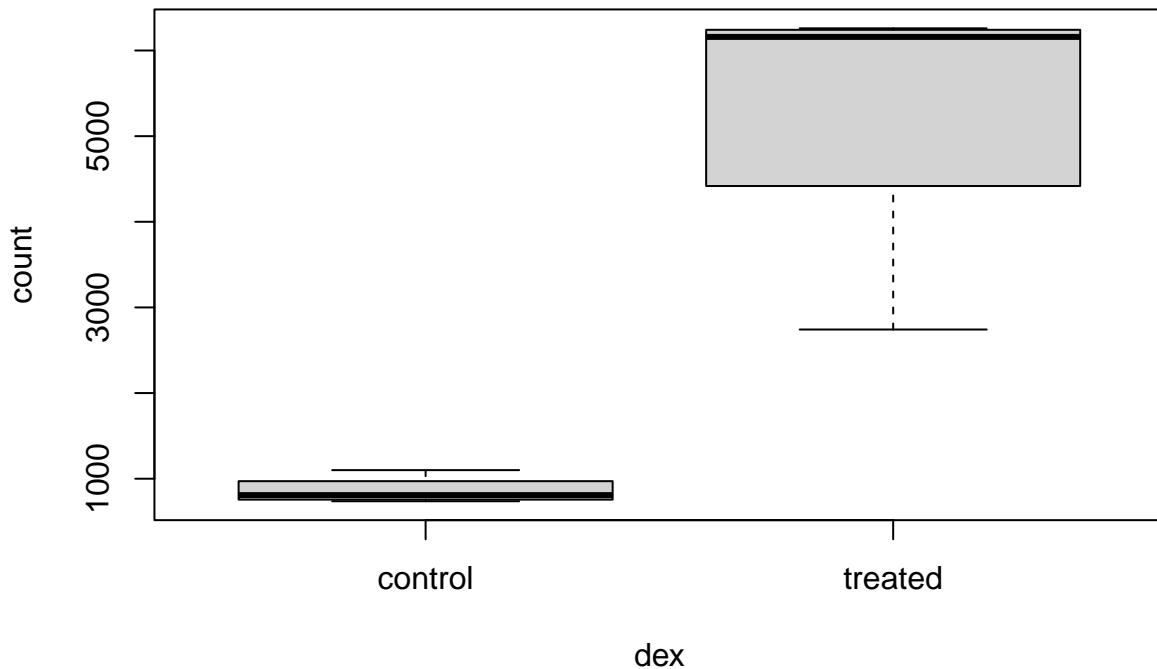
ENSG00000103196



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

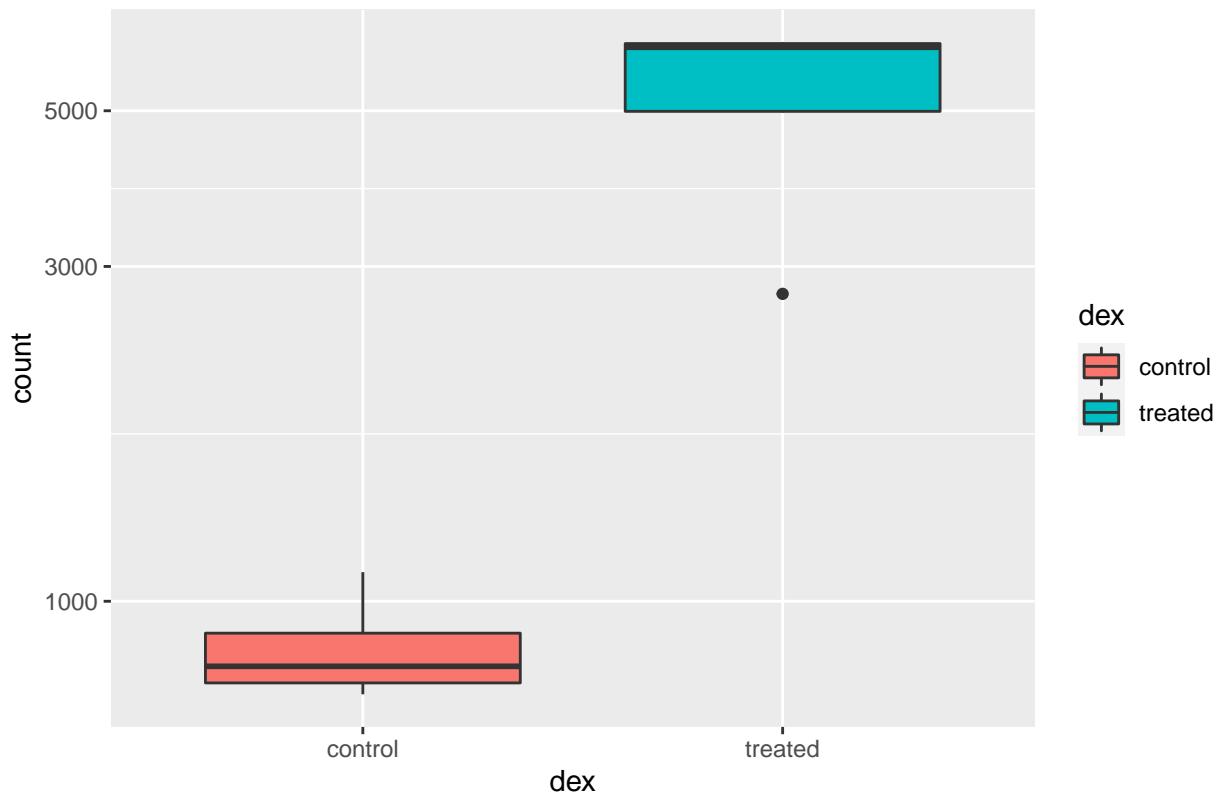
```
##           count     dex
## SRR1039508 774.5002 control
## SRR1039509 6258.7915 treated
## SRR1039512 1100.2741 control
## SRR1039513 6093.0324 treated
## SRR1039516  736.9483 control
## SRR1039517 2742.1908 treated
```

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



```
#Or do it all on ggplot2
library(ggplot2)
ggplot(d, aes(dex, count, fill=dex)) +
  geom_boxplot() +
  scale_y_log10() +
  ggtitle("CRISPLD2")
```

CRISPLD2



```
#Session info check
```

```
sessionInfo()
```

```
## R version 4.1.1 (2021-08-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Catalina 10.15.7
##
## Matrix products: default
## BLAS:    /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.0.dylib
## LAPACK:  /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats4      stats       graphics   grDevices  utils      datasets   methods
## [8] base
##
## other attached packages:
##  [1] gageData_2.32.0          gage_2.44.0
##  [3] pathview_1.34.0          org.Hs.eg.db_3.14.0
##  [5] AnnotationDbi_1.56.2     DESeq2_1.34.0
##  [7] SummarizedExperiment_1.24.0 Biobase_2.54.0
##  [9] MatrixGenerics_1.6.0      matrixStats_0.61.0
## [11] GenomicRanges_1.46.0      GenomeInfoDb_1.30.0
```

```

## [13] IRanges_2.28.0           S4Vectors_0.32.2
## [15] BiocGenerics_0.40.0      ggplot2_3.3.5
##
## loaded via a namespace (and not attached):
## [1] httr_1.4.2                 bit64_4.0.5          splines_4.1.1
## [4] highr_0.9                  blob_1.2.2            GenomeInfoDbData_1.2.7
## [7] yaml_2.2.1                 pillar_1.6.4          RSQLite_2.2.8
## [10] lattice_0.20-45           glue_1.5.0            digest_0.6.28
## [13] RColorBrewer_1.1-2        XVector_0.34.0         colorspace_2.0-2
## [16] htmltools_0.5.2           Matrix_1.3-4          XML_3.99-0.8
## [19] pkgconfig_2.0.3            genefilter_1.76.0       zlibbioc_1.40.0
## [22] GO.db_3.14.0              purrr_0.3.4            xtable_1.8-4
## [25] scales_1.1.1              BiocParallel_1.28.0     tibble_3.1.6
## [28] annotate_1.72.0           KEGGREST_1.34.0         generics_0.1.1
## [31] farver_2.1.0               ellipsis_0.3.2          cachem_1.0.6
## [34] withr_2.4.2                survival_3.2-13         magrittr_2.0.1
## [37] crayon_1.4.2               KEGGgraph_1.54.0         memoise_2.0.0
## [40] evaluate_0.14              fansi_0.5.0             graph_1.72.0
## [43] tools_4.1.1                lifecycle_1.0.1         stringr_1.4.0
## [46] locfit_1.5-9.4             munsell_0.5.0           DelayedArray_0.20.0
## [49] Biostrings_2.62.0           compiler_4.1.1          rlang_0.4.12
## [52] grid_4.1.1                 RCurl_1.98-1.5          bitops_1.0-7
## [55] labeling_0.4.2              rmarkdown_2.11           gtable_0.3.0
## [58] DBI_1.1.1                  R6_2.5.1                knitr_1.36
## [61] dplyr_1.0.7                 fastmap_1.1.0           bit_4.0.4
## [64] utf8_1.2.2                  Rgraphviz_2.38.0         stringi_1.7.5
## [67] parallel_4.1.1              Rcpp_1.0.7                vctrs_0.3.8
## [70] geneplotter_1.72.0          png_0.1-7                tidyselect_1.1.1
## [73] xfun_0.28

```