

Transcriptomics and the analysis of RNA-Seq data

Vera Sophia Beliaev

2/26/2022

1. Bioconductor and DESeq2 setup

Bioconductor and SESeq2 were successfully installed

Alignment Procedure: FASTQ files (contain nt seq and quality scores) aligned to reference and output stored in SAM/BAM format. Then tools must be used to count reads and assign them to genomic features (SAM/BAM files + GFF3 or GTF file) - Ex. kallisto, txlimport. End result is a count matrix that indicates number of reads aligned to a specific gene (rows) for each sample (columns)

DESeq2 Required Inputs: 1. a data.frame of countData (count matrix) and 2. a data.frame with sample metadata i.e. colData (data about the columns i.e. samples), The names of the samples in each data.frame should match, DESeq2 takes in un-normalized counts best

Count matrix: rows can be genes, binding regions (ChIP-Seq), peptide sequences (quantitative mass spec)

2. Import countData and colData

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

```
head(counts)
```

```
##          SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG000000000003      723       486       904       445      1170
## ENSG000000000005        0         0         0         0         0
## ENSG00000000419       467       523       616       371      582
## ENSG00000000457       347       258       364       237      318
## ENSG00000000460        96        81        73        66      118
## ENSG00000000938        0         0         1         0         2
##          SRR1039517 SRR1039520 SRR1039521
## ENSG000000000003     1097       806       604
## ENSG000000000005        0         0         0
## ENSG00000000419       781       417       509
## ENSG00000000457       447       330       324
## ENSG00000000460        94       102        74
## ENSG00000000938        0         0         0
```

```
head(metadata)

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

Q1. How many genes are in this dataset? There are 38,694 genes in the dataset.

```
nrow(counts)
```

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

Q2. How many ‘control’ cell lines do we have? There are 4 control cell lines.

```
View(metadata)
```

```
sum(with(metadata, dex == "control"))
```

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

```
#Another way of counting:
sum(metadata$dex == "control")
```

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

3. Toy differential expression

Note: this section is for control purposes only

```
#This code finds the sample id of control cell lines and calculates the mean counts per gene for those .
control <- metadata[metadata[, "dex"]== "control",]
control.counts <- counts[, control$id]
control.mean <- rowSums( control.counts )/4
head(control.mean)
```

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

```
#This is doing the same thing but with the dplyr package
library(dplyr)
```

```

## 
## Attaching package: 'dplyr'

## 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
## ENSG0000000938
##           0.75

```

Q3. How would you make the above code in either approach more robust? The division by 4 in the code is specific only to this data. This should be replaced with more general code that counts the number of control cell types in the data.frame for the calculation.

Q4. Follow the same procedure for the treated samples (i.e. calculate the mean per gene across drug treated samples and assign to a labeled vector called treated.mean)

```

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

## ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
##           658.00          0.00         546.00        316.50         78.75
## ENSG0000000938
##           0.00

#Combine meancount data
meancounts <- data.frame(control.mean, treated.mean)

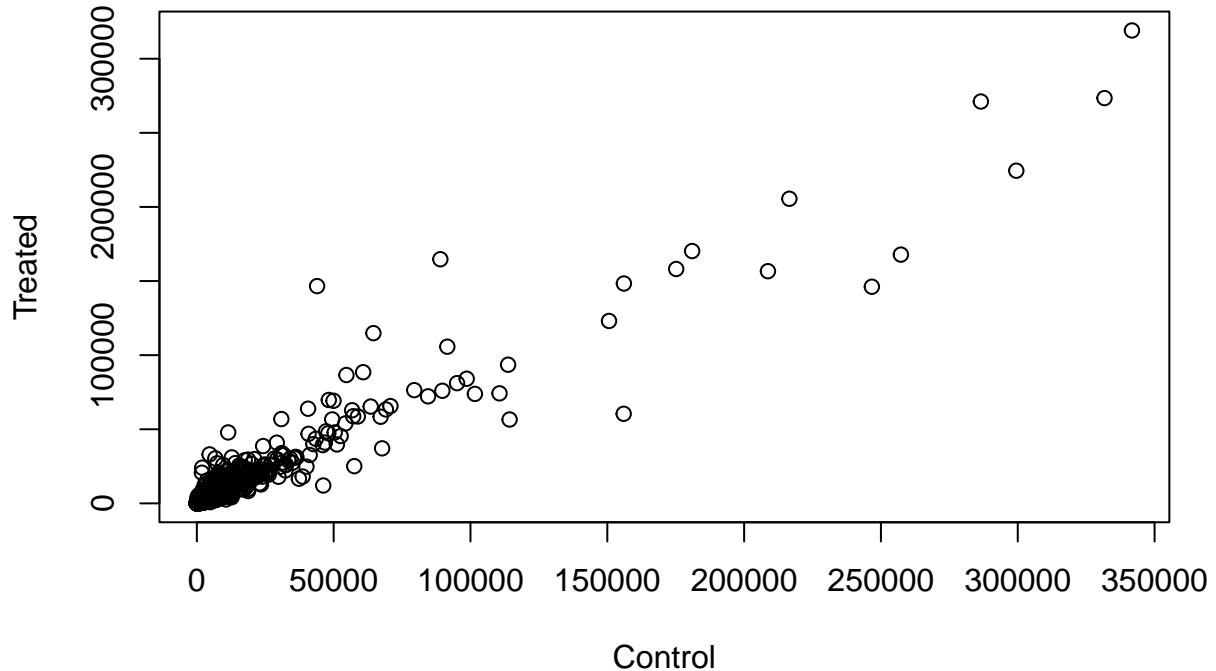
colSums(meancounts)

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

```

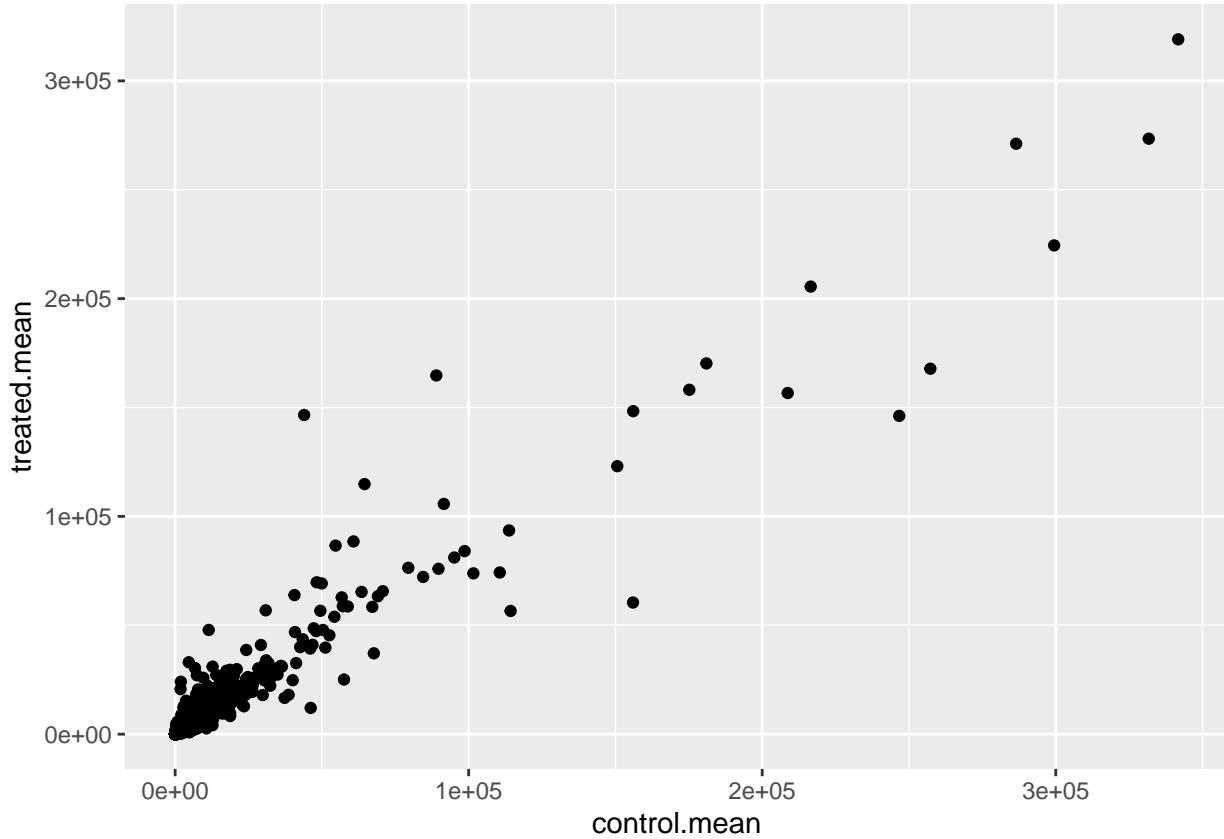
Q5 (a). Create a scatter plot showing the mean of the treated samples against the mean of the control samples. Your plot should look something like the following.

```
#Creating a scatter plot that shows the means against one another
plot(control.mean, treated.mean, 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? To make the plot, you use the geom_point() function.

```
#Make the plot using ggplot2
library(ggplot2)
ggplot(meancounts, aes(x=control.mean, y=treated.mean)) + geom_point()
```

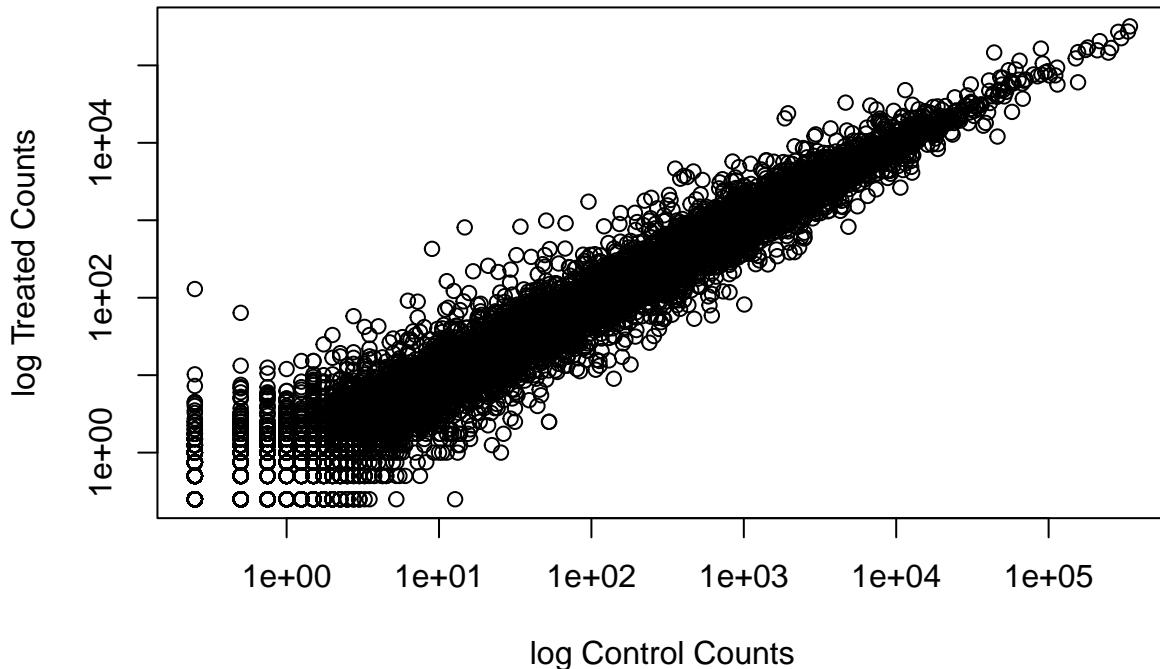


Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

```
plot(control.mean, treated.mean, xlab= "log Control Counts", ylab= "log Treated Counts", 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 value will be zero when there is no change. Doubling gives log2=1, quadruple gives log2=2, halved gives log2=1/2, etc

```
#Create new column with transformation
meancounts$log2fc <- log2(meancounts$treated / meancounts$control)
```

```
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

Now we need to remove results that are “-Inf” or infinity, of “NaN” or not a number.

```
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.ind argument in the which() function will return the indices of those rows which are TRUE for having meancounts in columns 1 and 2 equal to zero. This output gives the row index in the first column and the column index in the second column. The unique() function ensures no row is duplicated before removing them.

Now let's see which genes are up or downregulated using the common threshold where log2(foldchange) is greater than 2 or less than -2

```

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

```
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 at the less than -2 fc level

```
sum(down.ind)
```

```
## [1] 367
```

Q10. Do you trust these results? Why or why not? There is no indication of significance at this point, so no.

4. DESeq2 analysis

```
library(DESeq2)
```

```

## Loading required package: S4Vectors

## Loading required package: stats4

## Loading required package: BiocGenerics

```

```

## 
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:dplyr':
## 
##     combine, intersect, setdiff, union

## 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:dplyr':
## 
##     first, rename

## The following objects are masked from 'package:base':
## 
##     expand.grid, I, unname

## Loading required package: IRanges

## 
## Attaching package: 'IRanges'

## The following objects are masked from 'package:dplyr':
## 
##     collapse, desc, slice

## The following object is masked from 'package:grDevices':
## 
##     windows

## Loading required package: GenomicRanges

## Loading required package: GenomeInfoDb

## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

```

```

## Loading required package: matrixStats

##
## Attaching package: 'matrixStats'

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

##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnyNs, 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, rowAnyNs, 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

```

DESeq works on a DESeqDataSet which contains 1. a count matrix, 2. a metadata file, 3. a formula that indicates the experiment design (design formula) Design formula tells which cols in colData (has sample metadata) specify experimental design and how these factors should be used in analysis i.e. the formula shows how the counts for each gene depend on the vars in colData

```

#Create the DESeqDataSet
# dex is specified as part of experimental design with the tilde ~
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): ENSG000000000003 ENSG000000000005 ... ENSG00000283120
##   ENSG00000283123
## rowData names(0):
## colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
## colData names(4): id dex celltype geo_id

```

Now we will run DESeq on that DESeqDataSet

```

#Run DESeq and assign the result to the same var
dds <- DESeq(dds)

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

#Now we look at the results of DESeq
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

```

```

## ENSG00000000005 0.0000 NA NA NA NA
## ENSG00000000419 520.1342 0.2061078 0.101059 2.039475 0.0414026
## ENSG00000000457 322.6648 0.0245269 0.145145 0.168982 0.8658106
## ENSG00000000460 87.6826 -0.1471420 0.257007 -0.572521 0.5669691
## ... ...
## ENSG0000283115 0.000000 NA NA NA NA
## ENSG0000283116 0.000000 NA NA NA NA
## ENSG0000283119 0.000000 NA NA NA NA
## ENSG0000283120 0.974916 -0.668258 1.69456 -0.394354 0.693319
## ENSG0000283123 0.000000 NA NA NA NA
## padj
## <numeric>
## ENSG00000000003 0.163035
## ENSG00000000005 NA
## ENSG00000000419 0.176032
## ENSG00000000457 0.961694
## ENSG00000000460 0.815849
## ...
## ENSG0000283115 NA
## ENSG0000283116 NA
## ENSG0000283119 NA
## ENSG0000283120 NA
## ENSG0000283123 NA

```

Let's make the results easier to view by converting the results to a data.frame before viewing

```
head(as.data.frame(res))
```

	baseMean	log2FoldChange	lfcSE	stat	pvalue
## ENSG00000000003	747.1941954	-0.35070302	0.1682457	-2.0844697	0.03711747
## ENSG00000000005	0.0000000	NA	NA	NA	NA
## ENSG00000000419	520.1341601	0.20610777	0.1010592	2.0394752	0.04140263
## ENSG00000000457	322.6648439	0.02452695	0.1451451	0.1689823	0.86581056
## ENSG00000000460	87.6826252	-0.14714205	0.2570073	-0.5725210	0.56696907
## ENSG00000000938	0.3191666	-1.73228897	3.4936010	-0.4958463	0.62000288
## padj					
## ENSG00000000003	0.1630348				
## ENSG00000000005	NA				
## ENSG00000000419	0.1760317				
## ENSG00000000457	0.9616942				
## ENSG00000000460	0.8158486				
## 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

#here, by default alpha=0.1

#adjust alpha value
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

```

5. Adding annotation data

```

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] "ACNUM"        "ALIAS"         "ENSEMBL"        "ENSEMLPROT"    "ENSEMLTRANS"
##   [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 add individ cols to our results table. we use row names as a key, specified keytype as ensembl, col
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

#now you can see the symbol column added onto our results
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
## ENSG000000000419 520.134160  0.2061078  0.101059  2.039475 0.0414026
## ENSG000000000457 322.664844  0.0245269  0.145145  0.168982 0.8658106
## ENSG000000000460 87.682625 -0.1471420  0.257007 -0.572521 0.5669691
## ENSG000000000938 0.319167 -1.7322890  3.493601 -0.495846 0.6200029
##          padj      symbol
##          <numeric> <character>
## ENSG00000000003  0.163035   TSPAN6
## ENSG00000000005      NA      TNMD
## ENSG000000000419  0.176032   DPM1
## ENSG000000000457  0.961694   SCYL3
## ENSG000000000460  0.815849   C1orf112
## ENSG000000000938      NA      FGR

```

Q11. Run the mapIds() function two more times to add the Entrez ID and UniProt accession and GENENAME as new columns called res\$entrez, res\$uniprot and res\$genename.

Note: column arg specifies which info to pull from ensembl Note: dataf\$NewColumn <- values creates new column in data frame

```

res$entrez <- mapIds(org.Hs.eg.db,
                      keys=row.names(res), # Our genenames
                      keytype="ENSEMBL",      # The format of our genenames
                      column="ENTREZID",      # The new format we want to add
                      multiVals="first")

```

```

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

```

```

res$uniprot <- mapIds(org.Hs.eg.db,
                      keys=row.names(res), # Our genenames
                      keytype="ENSEMBL",      # The format of our genenames
                      column="UNIPROT",      # The new format we want to add
                      multiVals="first")

```

```

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

```

```

res$genename <- mapIds(org.Hs.eg.db,
                      keys=row.names(res), # Our genenames
                      keytype="ENSEMBL",      # The format of our genenames
                      column="GENENAME",      # 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 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, ..
## Order the results by adj p value
ord <- order( res$padj )
#View(res[ord,])
head(res[ord,])

## log2 fold change (MLE): dex treated vs control
## Wald test p-value: dex treated vs control
## DataFrame with 6 rows and 10 columns
##           baseMean log2FoldChange      lfcSE      stat     pvalue
##           <numeric>      <numeric> <numeric> <numeric> <numeric>
## ENSG00000152583  954.771      4.36836  0.2371268  18.4220 8.74490e-76
## ENSG00000179094  743.253      2.86389  0.1755693  16.3120 8.10784e-60
## ENSG00000116584  2277.913     -1.03470  0.0650984 -15.8944 6.92855e-57
## ENSG00000189221  2383.754      3.34154  0.2124058  15.7319 9.14433e-56
## ENSG00000120129  3440.704      2.96521  0.2036951  14.5571 5.26424e-48
## ENSG00000148175  13493.920     1.42717  0.1003890  14.2164 7.25128e-46
##          padj      symbol      entrez      uniprot
##          <numeric> <character> <character> <character>
## ENSG00000152583  1.32441e-71    SPARCL1      8404 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

#show ordered sig results w/ annotations
write.csv(res[ord,], "deseq_results.csv")

```

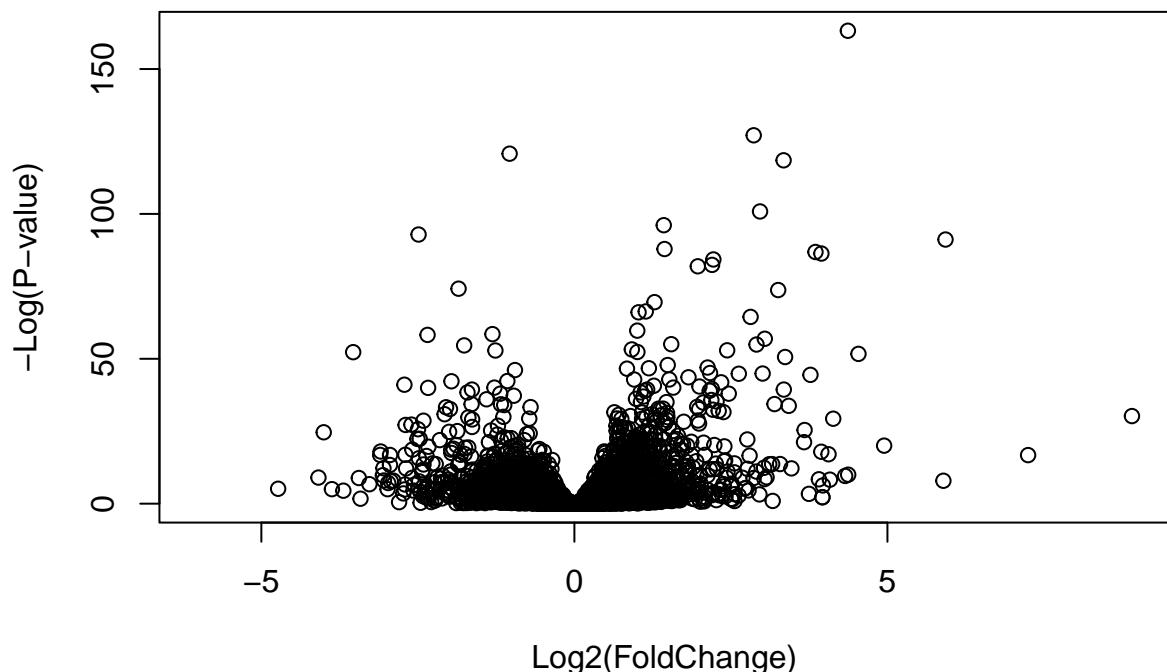
6. Data Visualization

Volcano Plots

```

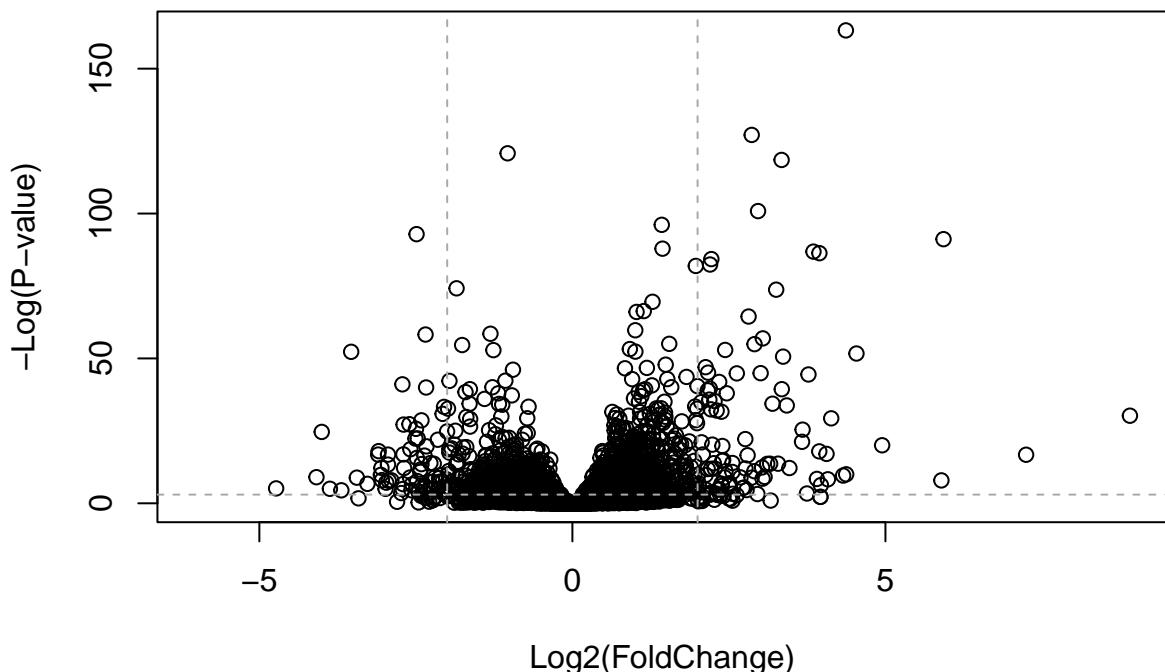
#simple volcano plot
plot( res$log2FoldChange, -log(res$padj),
      xlab="Log2(FoldChange)",
      ylab="-Log(P-value)")

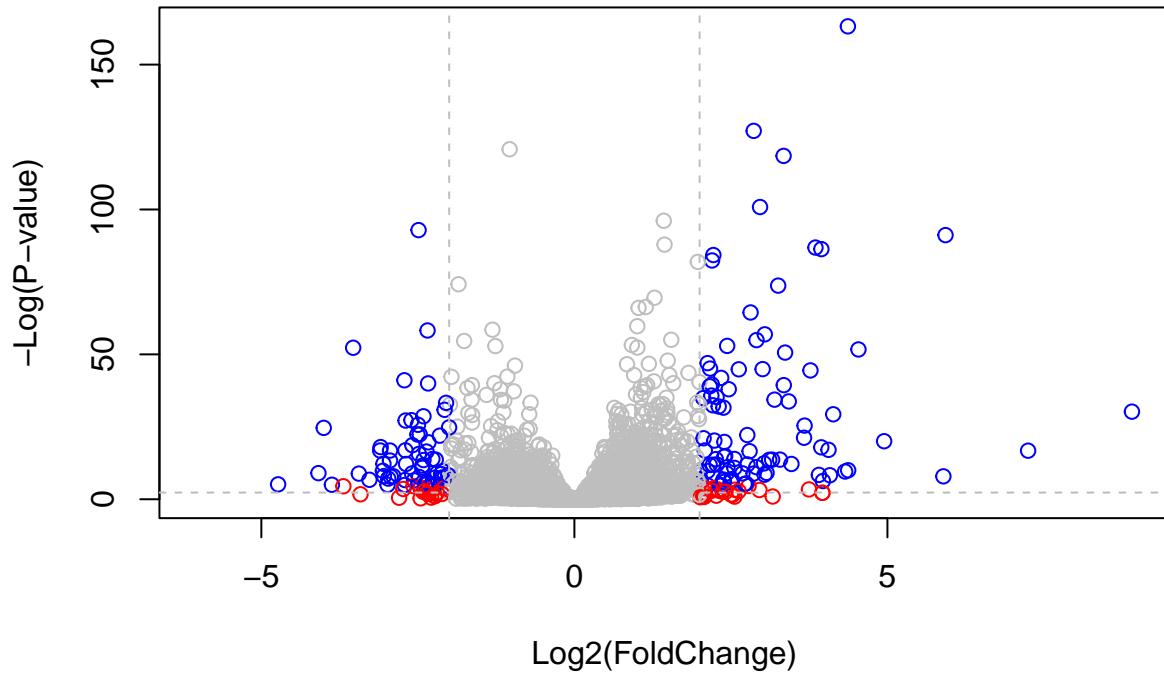
```



```
#use abline() and color to highlight genes that have padj<0.05 and the abs log2FoldChange>2
plot( res$log2FoldChange, -log(res$padj),
      ylab="-Log(P-value)", xlab="Log2(FoldChange)")

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





The EnhancedVolcano package can be useful

```
library(EnhancedVolcano)

## Loading required package: ggrepel

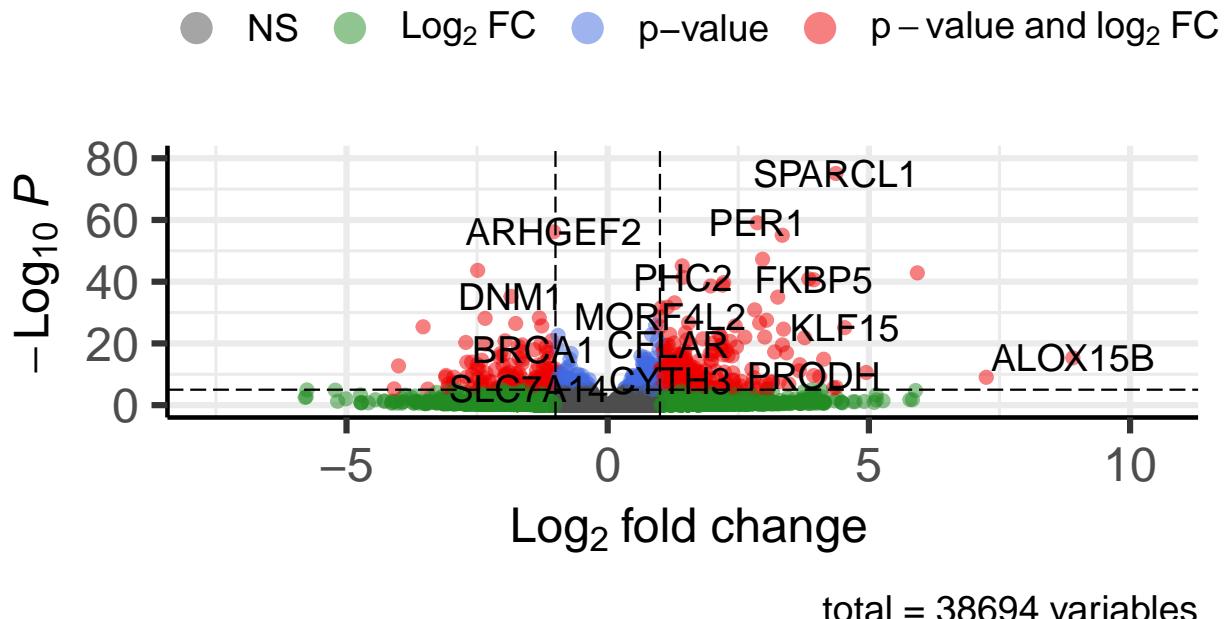
## Registered S3 methods overwritten by 'ggalt':
##   method           from
##   grid.draw.absoluteGrob  ggplot2
##   grobHeight.absoluteGrob ggplot2
##   grobWidth.absoluteGrob ggplot2
##   grobX.absoluteGrob     ggplot2
##   grobY.absoluteGrob     ggplot2

#Volcano plot with most interesting genes labeled by name
x <- as.data.frame(res)

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

Volcano plot

EnhancedVolcano



7. Pathway Analysis

Here we will use the GAGE package and the KEGG pathway database for KEGG pathway enrichment analysis. The pathview package will draw pathway diagrams

```
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)

#kegg.sets.hs is a list of 229 elements, e/ elem is a character vect of entrez ids for a KEGG pathway.

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"

```

gage() fn requires 1. a named vector of fold changes (ours are stored from DESeq2 analysis in res) and 2. Entrez gene IDs (ours are stored in a made res column)

```

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, the rows are sorted by p-value
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

```

```

#make a pathway plot with the pathview package, downloads pathway difure data from KEGG and adds our re
pathview(gene.data=foldchanges, pathway.id="hsa05310")

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

## Info: Working in directory C:/Users/belia/OneDrive/Desktop/BIMM143/Transcriptomics

## 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 C:/Users/belia/OneDrive/Desktop/BIMM143/Transcriptomics

## 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?

# Look at the first 2 up (greater) pathways
head(keggres$greater, 2)

##                                     p.geomean stat.mean      p.val
## hsa00500 Starch and sucrose metabolism 0.003306262 2.772644 0.003306262
## hsa00330 Arginine and proline metabolism 0.012317455 2.280002 0.012317455
##                                     q.val set.size      exp1
## hsa00500 Starch and sucrose metabolism 0.7042337      52 0.003306262
## hsa00330 Arginine and proline metabolism 0.7774866      54 0.012317455

pathview(gene.data=foldchanges, pathway.id="hsa00500 ")

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

## Info: Working in directory C:/Users/belia/OneDrive/Desktop/BIMM143/Transcriptomics

## Info: Writing image file hsa00500 .pathview.png

## Info: some node width is different from others, and hence adjusted!

pathview(gene.data=foldchanges, pathway.id="hsa00330")

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

## Info: Working in directory C:/Users/belia/OneDrive/Desktop/BIMM143/Transcriptomics

## Info: Writing image file hsa00330.pathview.png

```

Plotting counts for genes of interest

DESeq2 has plotCounts() which takes 1. DESeqDataSet, 2. gene name, 3. var from colData and makes a plot

```
#find gene ID for CRISPLD2 gene
i <- grep("CRISPLD2", res$symbol)
res[i,]

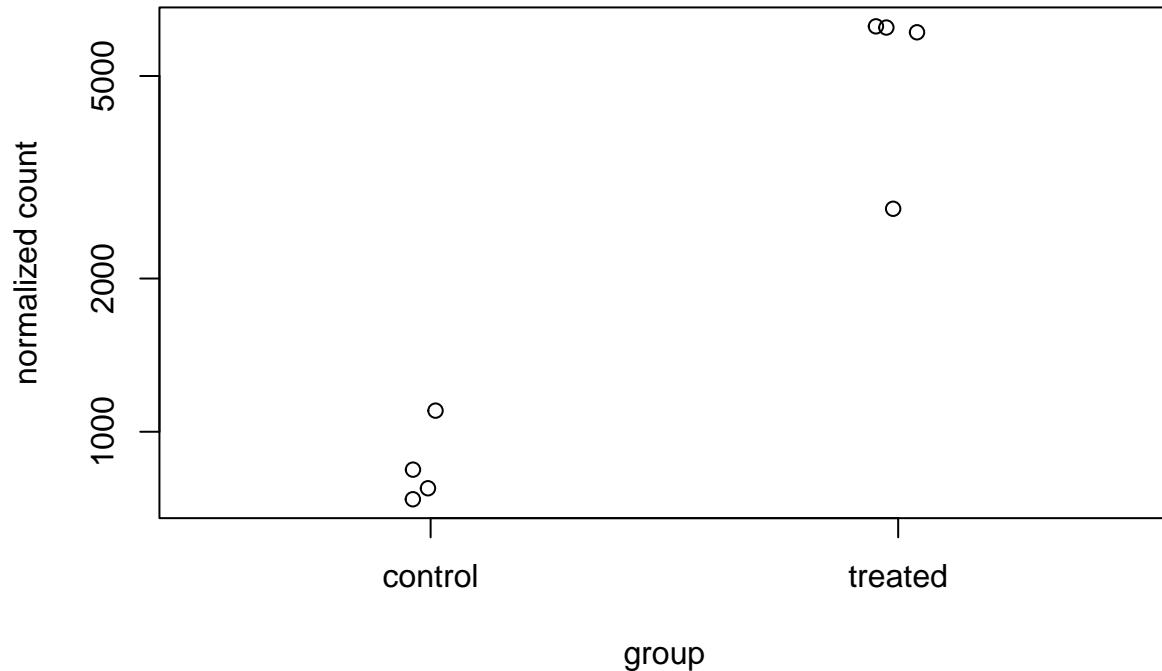
## log2 fold change (MLE): dex treated vs control
## Wald test p-value: dex treated vs control
## DataFrame with 1 row and 10 columns
##           baseMean log2FoldChange      lfcSE      stat      pvalue
##           <numeric>     <numeric> <numeric> <numeric>    <numeric>
## ENSG00000103196   3096.16      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"

#plot the counts
plotCounts(dds, gene="ENSG00000103196", intgroup="dex")
```

ENSG00000103196

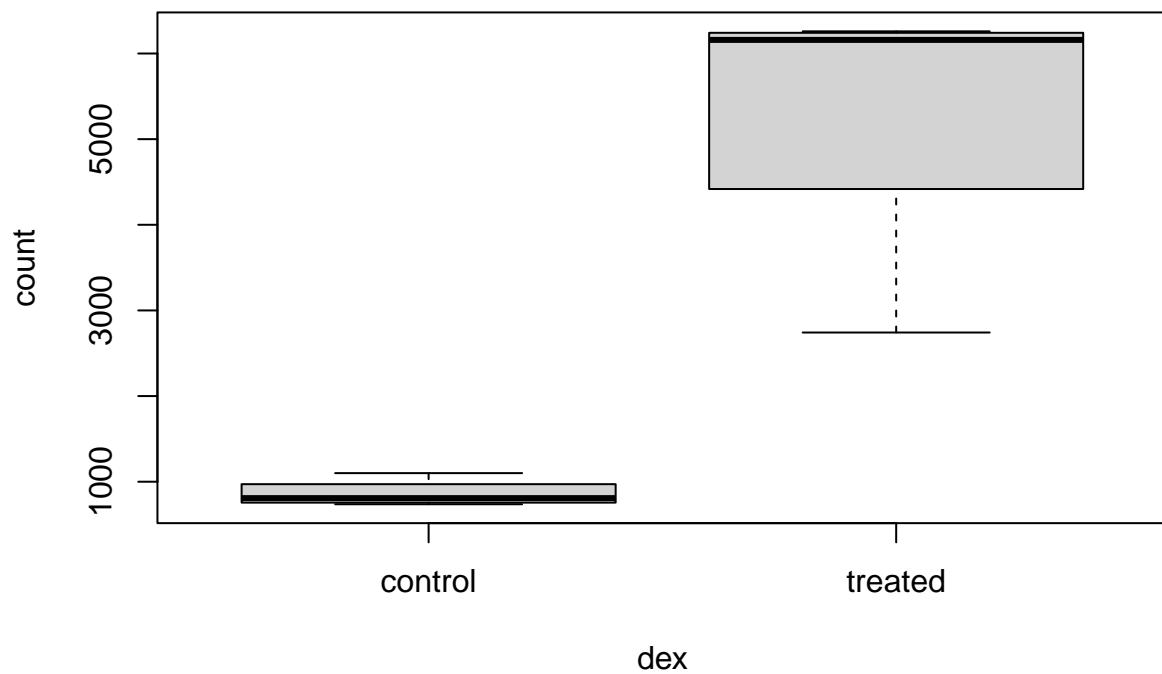


Let's take the data from this and use it in ggplot

```
# 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

