Class12

Jinsung

Transcriptomics and the analysis of RNA-Seq data

Bioconductor and DESeq2 setup

Bioconductor packages are installed differently than "regular" R packages from CRAN. WE need to setup the appropriate tool in order to process the data in the way we want.

install.packages("BiocManager") BiocManager::install() BiocManager::install("DESeq2")

To check the successful installment, use 'library()'

```
library(BiocManager)
```

```
Warning: package 'BiocManager' was built under R version 4.2.2
```

```
Bioconductor version '3.15' is out-of-date; the current release version '3.16'
  is available with R version '4.2'; see https://bioconductor.org/install
```

```
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


Attaching package: 'IRanges'

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
```

```
Warning: package 'matrixStats' was built under R version 4.2.2


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
```

Since we didn't get error message, it means the packages are installed correctly!

RNA-Seq analysis begins from FASTQ files that contain nucleotide sequence and quality score of each position.They must be alinged with a reference, which gets stored as SAM/BAM.

In our work, the transcripted was quantified using 'kallisto' then summarized with 'txlmport, which is suitable for analysis by DESeq.

DESeq2 package expects input of first data frame of 'count data' and second data frame of sample 'metadata'(or colData).

Import countData and colData

First, we read off the given csv file data.

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

There are 38694 genes are in the data set.

Before moving on, we should check correspondence of metadata and count data

```
metadata$id
```

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

```
colnames(counts)
```

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

To check their orders, we can use '==' test of equality.It returns logical value.

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

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

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

```
[1] TRUE
```

'all()' let's us see if all element in the vector is the same. All true! Seems about right now.

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

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

```
control treated
      4       4
```

There are 4 control cell lines in the data set.

or to find the location of 'control' and specificity,

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

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

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

```
          id     dex celltype     geo_id
1 SRR1039508 control   N61311 GSM1275862
3 SRR1039512 control  N052611 GSM1275866
5 SRR1039516 control  N080611 GSM1275870
7 SRR1039520 control  N061011 GSM1275874
```

Now I can use this to access just the "control" columns of my 'counts' data.

```
control.counts <- counts[,control$id]
head(control.counts)
```

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

Find the mean count value for each gene by binding 'rowMeans()'.

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

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         900.75            0.00          520.50          339.75            97.25
ENSG00000000938
           0.75
```

It could also be done by using the dplyr package from the tidyverse.

```
library(dplyr)
```

```
Attaching package: 'dplyr'

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

    combine

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

    count

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

    intersect, setdiff, union

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

    intersect

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

    collapse, desc, intersect, setdiff, slice, union

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

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

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

    combine, intersect, setdiff, union

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

    filter, lag

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

    intersect, setdiff, setequal, union
```

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

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         900.75            0.00          520.50          339.75           97.25
ENSG00000000938
           0.75
```

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

We can simplify the code by:

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

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         900.75            0.00          520.50          339.75           97.25
ENSG00000000938
           0.75
```

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.id <- metadata[metadata$dex=="treated","id"]
treated.mean <- rowMeans(counts[ ,treated.id])
```

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

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

Now, let's put those two mean data together.

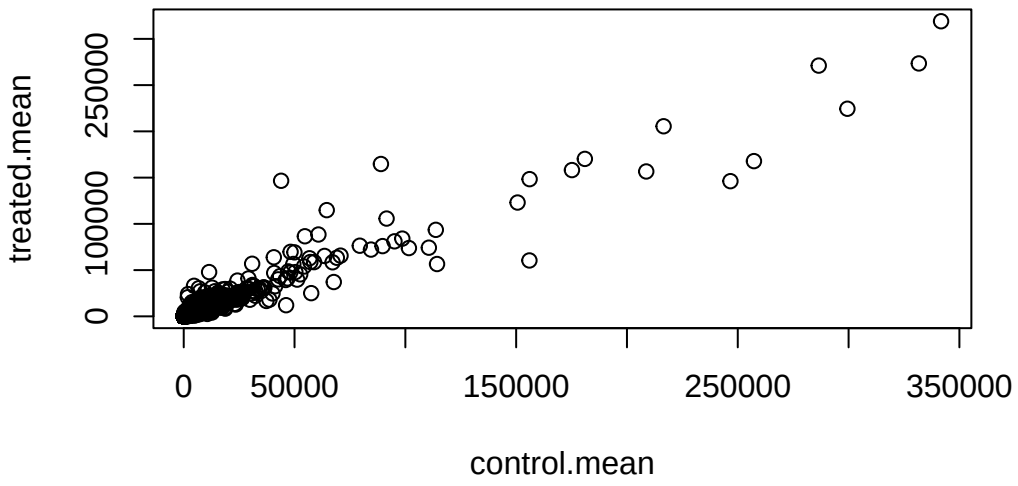
```
meancounts <- data.frame(control.mean, treated.mean)
head(meancounts)
```

```
                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.

Let's do quick plot to see the values.

```
plot(meancounts)
```



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

For ggplot version of this, we would use 'geom_point' for scatter plot.

This is heavily skewed and over a wide range. Needs modification!

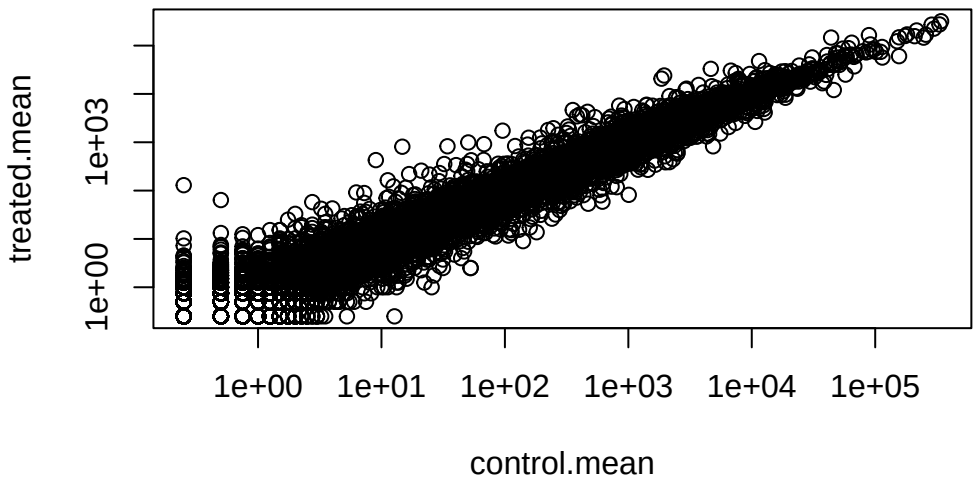
Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

The argument 'log' allows us to scale the graph differently.

```
plot(meancounts, log="xy")
```

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

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



Visualizing and transforming data with log makes interpretation easier. Especially log2 that can spot the changes from correlation.

Log2 fold change of our data woud be useful.

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

```
[1] -0.45303916         NaN  0.06900279 -0.10226805 -0.30441833        -Inf
```

The NaN is returned when you divide by zero and try to take the log. The -Inf is returned when you try to take the log of zero. So there are some zeros in the data that we need to filter out.

```
tokeep <- rowSums(meancounts[,1:2]==0) == 0
head(tokeep)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
           TRUE           FALSE            TRUE            TRUE            TRUE
ENSG00000000938
          FALSE
```

```
mycounts <- meancounts[tokeep,]
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 argument is to change logical vectors into numerical values. Unique() function finds the value '1' referring to TRUE in the data.

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

A common threshold for calling genes are differently expressed is a log fold change of +2 or -2

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

```
[1] 314
```

Out of data set, 314 are upregulated genes.

```
round(sum(mycounts$log2fc >= 2)/nrow(mycounts)*100, 2)
```

```
[1] 1.44
```

Percentage of upregulated gene is 1.44%

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

Number and proportion of downregulated genes would be:

```
sum(mycounts$log2fc <= -2)
```

```
[1] 485
```

```
round(sum(mycounts$log2fc <= -2)/nrow(mycounts)*100, 2)
```

```
[1] 2.22
```

Finding statistically significance of this drug induced difference would be the next step.

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

I trust this results because it was picked based on threshold point with log2 fold-change that determines significance of the difference.

DSEq2

```
library(DESeq2)
```

Main function in the DESeq2 package is called 'deseq()'. It wants the count data and metadata (colData) as input in a specific way.

```
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
```

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

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

```
results(dds)
```

```
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>
ENSG00000000003  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
...                    ...            ...       ...       ...       ...
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>
ENSG00000000003  0.163035
ENSG00000000005        NA
ENSG00000000419  0.176032
ENSG00000000457  0.961694
```

```
ENSG00000000460  0.815849
...                   ...
ENSG00000283115        NA
ENSG00000283116        NA
ENSG00000283119        NA
ENSG00000283120        NA
ENSG00000283123        NA
```

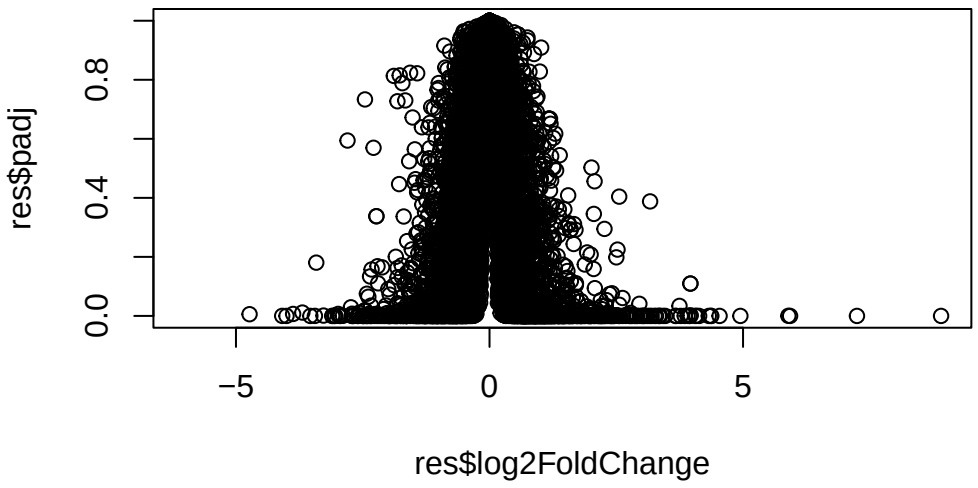
We have log2 fold-change and adjusted p value so far for the significance.

Data Visualization

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

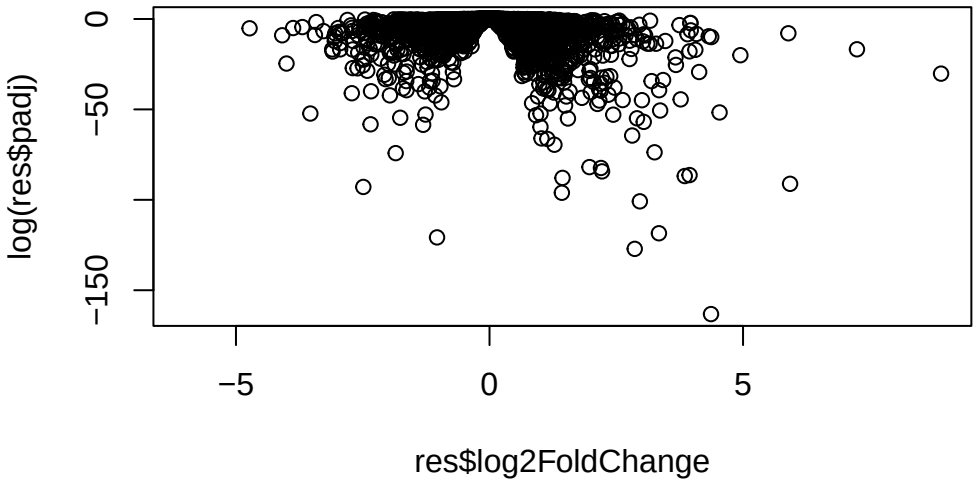
Let's first plot log2fc and adjusted p-value

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



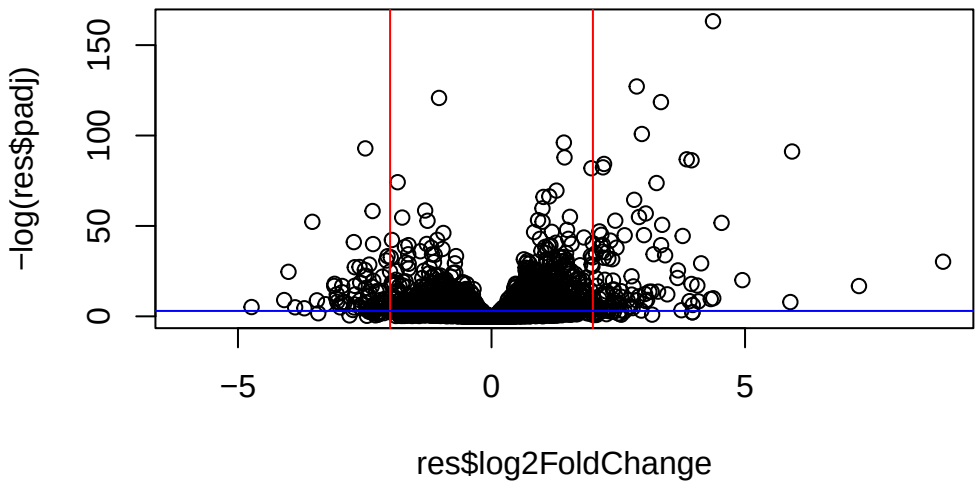
It is little unncessary because it includes bunch of data points with high p-value that we do not care about. Let's take log of p-value.

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



Points with the low log(res$padj) would be the points to care about but it is less intuitive to look at.We can flip the Y-axis so the plot would look better.

```
plot(res$log2FoldChange, -log(res$padj))
abline(v=c(-2, +2), col="red")
abline(h=-log(0.05), col="blue")
```



It is called volcano plot!

We are looking for data points with +2 log2 fold-change and high -log(padj) values, which are in the lined segments.

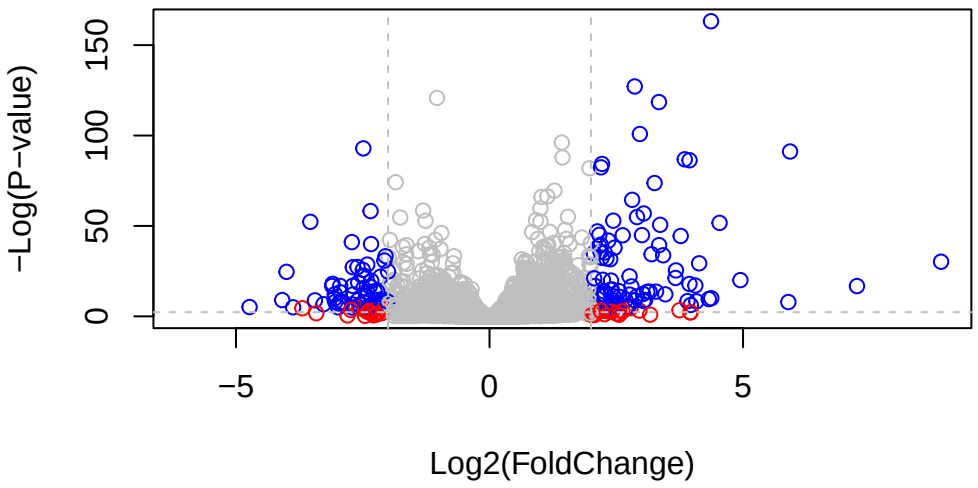
Let's make it colorful!

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

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

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

abline(v=c(-2,2), col="gray", lty=2)
abline(h=-log(0.1), col="gray", lty=2)
```



Annotation of Gene Results.

We start by intalling and looking the appropriate packages from the Bioc.

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

```
Attaching package: 'AnnotationDbi'

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

    select
```

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

'mapIDs()' function "maps" database identifiers between different databases, meaning it translates the identifiers used by one database to that used by another database.

Check the availability by looking through the column.

```
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"
```

Our input data is 'Ensembl', which we will translate into 'Symbol'

Our result data is saved as 'res'

```
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
```

We can assign the new values to 'res$symbol' using mapIDs.

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

```
                      column="SYMBOL",
                      multiVals="first")
```

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

```
  head(res$symbol)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
       "TSPAN6"          "TNMD"          "DPM1"         "SCYL3"      "C1orf112"
ENSG00000000938
          "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),
                       keytype="ENSEMBL",
                       column="ENTREZID",
                       multiVals="first")
```

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

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

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

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

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

```
head(res)
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 10 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
                 <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
                     padj      symbol      entrez     uniprot
                <numeric> <character> <character> <character>
ENSG00000000003  0.163035      TSPAN6        7105  A0A024RCI0
ENSG00000000005        NA        TNMD       64102      Q9H2S6
ENSG00000000419  0.176032        DPM1        8813      O60762
ENSG00000000457  0.961694       SCYL3       57147      Q8IZE3
ENSG00000000460  0.815849    C1orf112       55732  A0A024R922
ENSG00000000938        NA         FGR        2268      P09769
                           genename
                        <character>
ENSG00000000003          tetraspanin 6
ENSG00000000005            tenomodulin
ENSG00000000419 dolichyl-phosphate m..
ENSG00000000457 SCY1 like pseudokina..
ENSG00000000460 chromosome 1 open re..
ENSG00000000938 FGR proto-oncogene, ..
```

Pathway Analysis

Pahtway analysis aims to reduce the complexity of interpreting gene lists via mapping the listed genes to known biological pathways, processes and fu nctions.

Some major genesets include KEGG, GO, etc. We will use the 'gage' package for our first pathway analysis.

'BiocManager::install(c("pathview", "gage", "gageData"))' Installed and to be loaded.

We can look at the first two pathways once loaded.

```
library(pathview)
library(gage)
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"
```

The main 'gage()' function wants a vector containing measure of importance as input. For us that is fold-change. Vector needs to have ENTREZ ids as the names.

Vectors can have names, which is useful for keeping track of what they are and the values that correspond to them.

```
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
```

Now we run the anlysis

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

Let's find what's in this result.

```
attributes(keggres)
```

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

'gage' splits the result into 'greater' and 'less' for regulation status. We can look at the first three downregulated pathway results.

```
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
```

More detailed information can be found at these pathways by 'pathview()' function that take KEGG ID and our vector of importance.

Let's look at asthma pathway hsa05310.

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

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

```
Info: Working in directory C:/Users/Jinsung/Downloads/BIMM 143/Class12
```

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

```
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
```

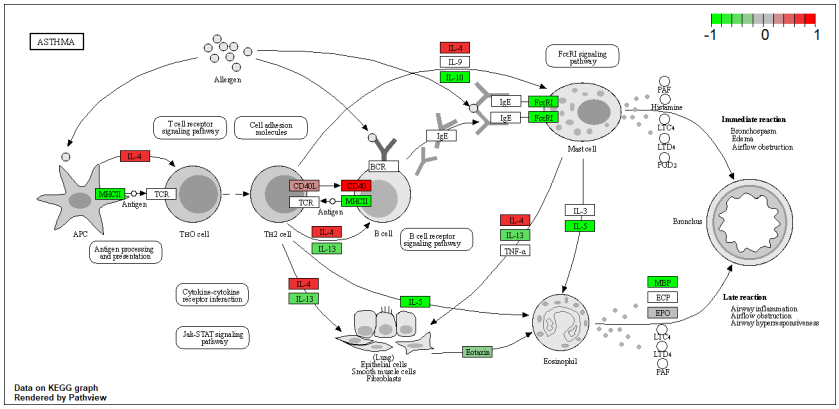


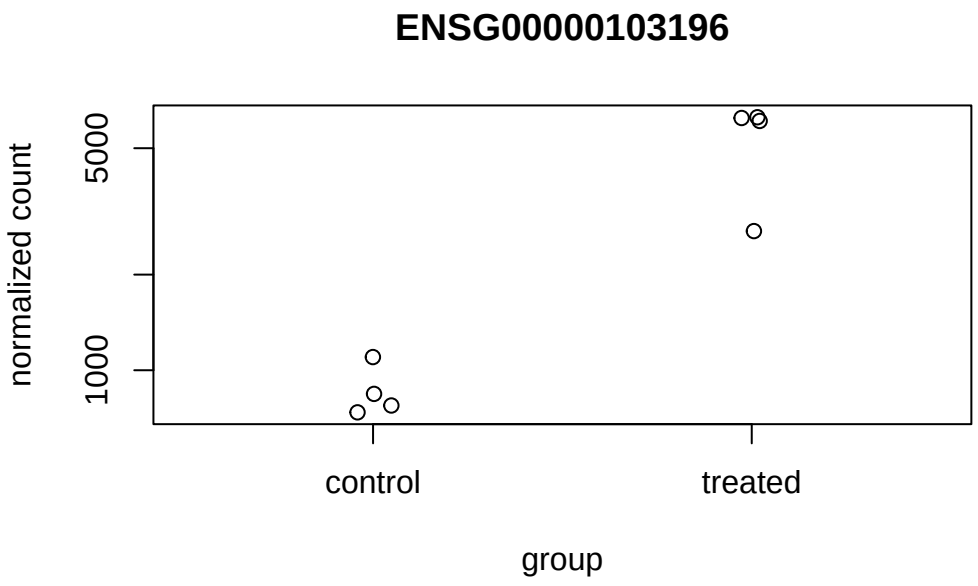
Figure 1: The Asthma Pathway with Our Genes

```
                     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"
```

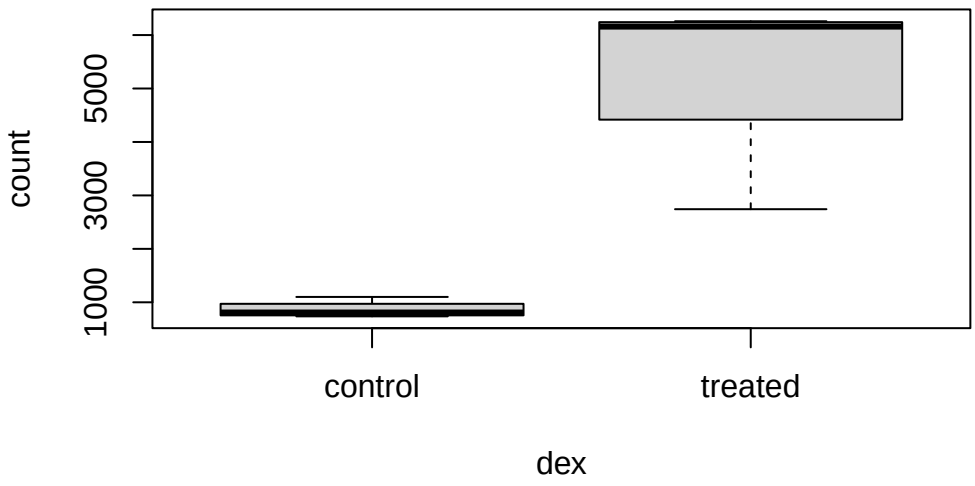
```
plotCounts(dds, gene="ENSG00000103196", intgroup="dex")
```



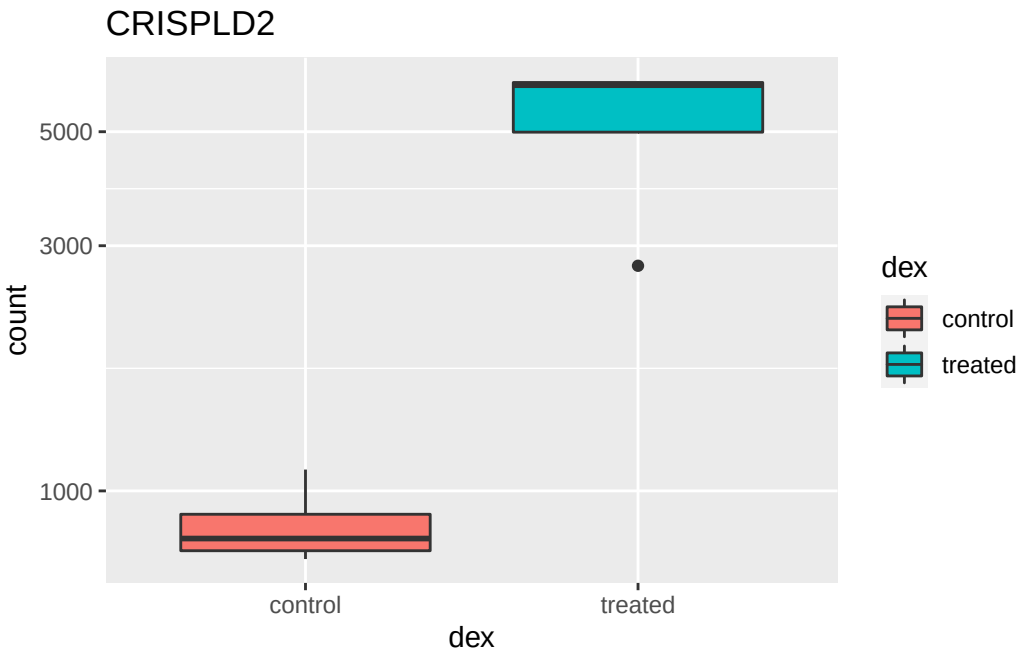

```
d <- plotCounts(dds, gene="ENSG00000103196", intgroup="dex", returnData=TRUE)
head(d)
```

```
               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