Transcriptomics and the analysis of RNA-Seq data

Adithi Kumar (PID: A16653979)

Bioconductor Setup

To install Bioconductor packages

1. #install.packages("BiocManager")
2. BiocManager::install(" ")

   We will install the DESeq2 bioconductor package with this format

```
library(BiocManager)
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, aperm, append, as.data.frame, basename, cbind,
    colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
    get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
    match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
    Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
    table, tapply, union, unique, unsplit, which.max, which.min


Attaching package: 'S4Vectors'

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

    findMatches

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

    expand.grid, I, unname

Loading required package: IRanges

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
```

The DESeq2 package requires 2 inputs: a data.frame of count data and a second data.frame with information about the samples (metadata or colData)

```
# Complete the missing code
counts <- read.csv("airway_scaledcounts.csv", row.names=1)
metadata <-  read.csv("airway_metadata.csv")
#Let's look at the datasets
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
```

The data for this hands-on session comes from a published TNA-seq experiment where airway smooth muscle cells were treated with **dexamethasone** (dex), a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al. 2014)

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

There are 38694 genes in the dataset.

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

```
table(metadata[,2])
```

```
control treated
      4       4
```

```
# Could also do: sum(metadata$dex == "control) or table(metadata$dex)
```

There are 4 'control' cell lines.

Toy Differential Gene Expression

Let's start by calculating the meant counts per gene in the control samples. We can then compare this value foe each gene to the mean counts in the treated samples (i.e. columns)

1. Find which columns in counts correspond to "control" samples
2. Calculate the mean value per gene in these columns
3. Store my answer for later in 'control.mean'

```
#make dataset with only controls
control <- metadata[metadata[,"dex"]=="control",]
#find IDs of each control
control.counts <- counts[ ,control$id]
# Fine Control means
control.mean <- rowSums( control.counts )/4
```

```
#Step 1: fine which cloumns in 'counts' correspond to control samples ( Another way to do
#control.ids <- metadata$dex == "control"

#metadata[controls.ids, ]

#control.count <- counts[counts.ids]
```

Q3. How would you make the above code in either approach more robust? Is there a function that could help here?

```
#apply (control.counts, 1, mean)
# OR
#rowMeans(control.counts)
```

The above code calculates control means by dividing by the set number of rows (4) so it wouldn't work if more samples were added; using the 'apply'function or 'rowMeans' would help alleviate this problem.

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
ENSG00000000938
           0.00
```

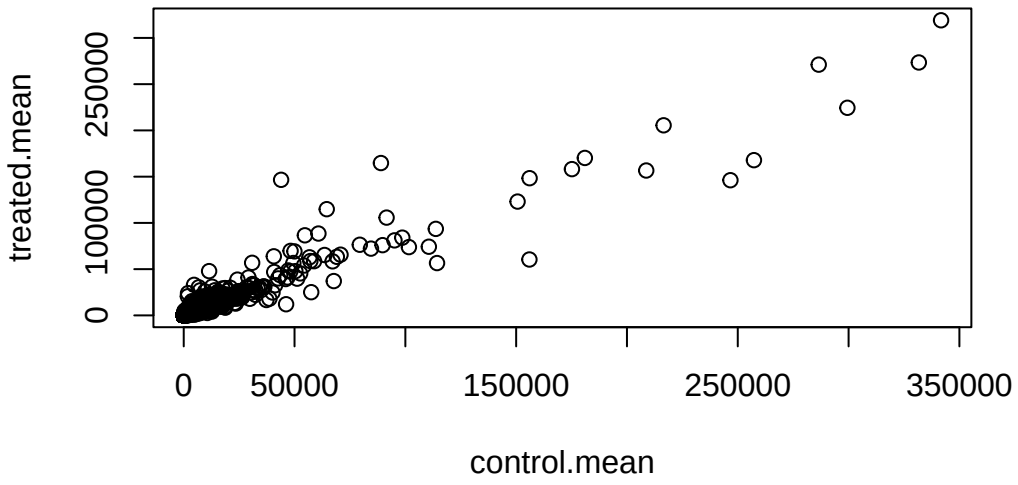
To keep us tidy, let's put 'control.mean' and 'treated.mean' vectors together as two columns of a new data.frame

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

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

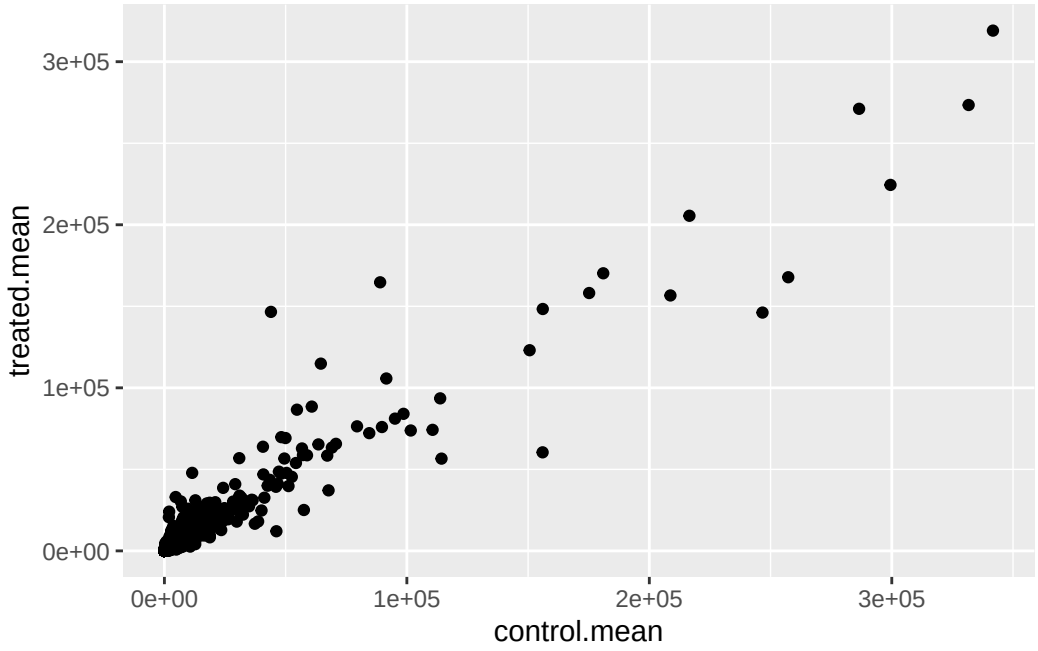
**** Q5 (a).** Create a scatter plot showing the mean of the treated samples against the mean of the control samples. Your plot should look something like the following.**

```
plot(meancounts)
```



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 = control.mean, y= treated.mean) +geom_point()
```



To make a scatterplot, use the 'geom_point' function.

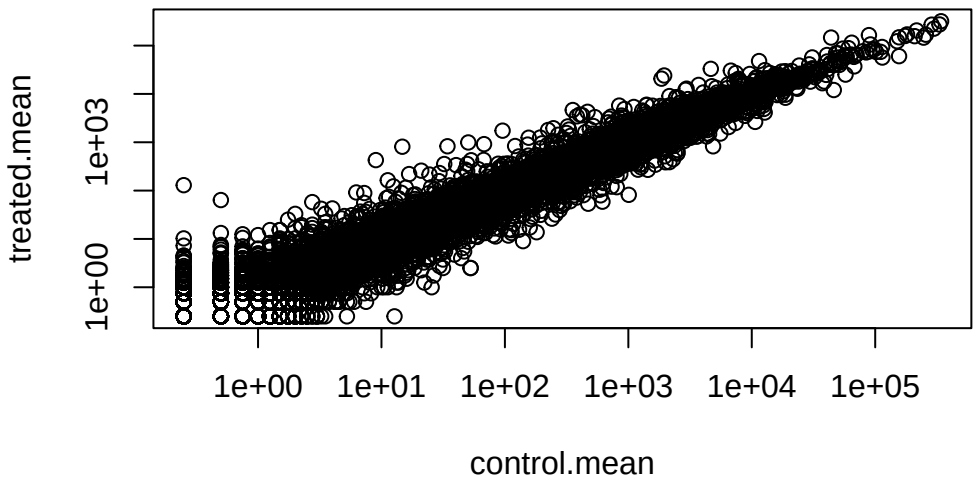
Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

In plot.default(), the log argument would plot both axes on a log scale.

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

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

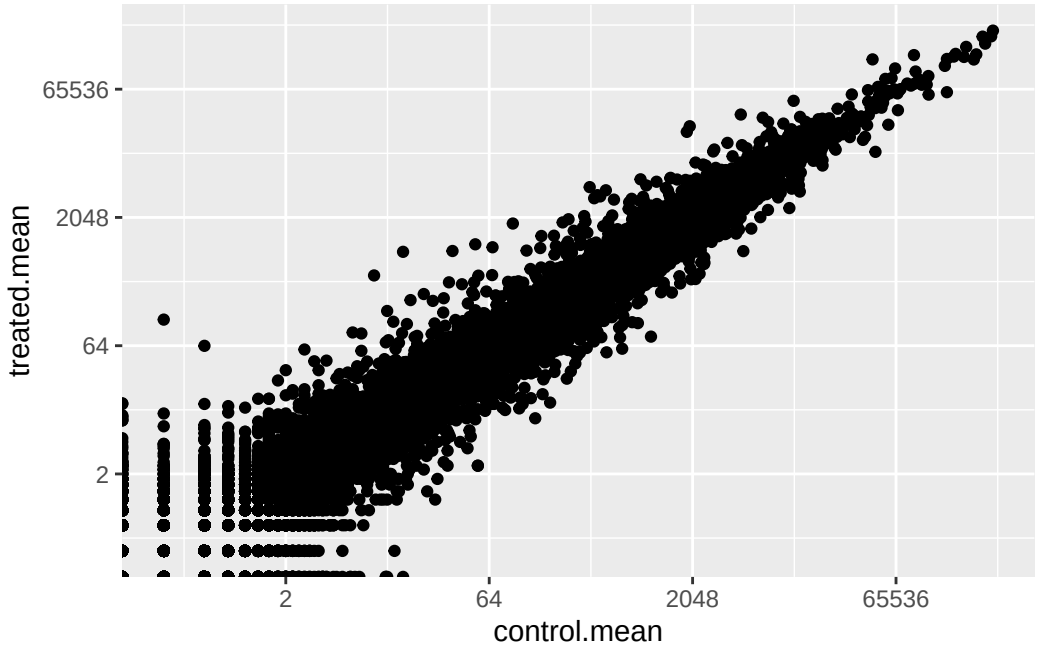
```
Warning in xy.coords(x, y, xlabel, ylabel, log): 15281 y values <= 0 omitted
from logarithmic plot
```




```
ggplot(meancounts) + aes(x = control.mean, y= treated.mean) +geom_point() +
scale_x_continuous(trans="log2") +
scale_y_continuous(trans="log2")
```

```
Warning: Transformation introduced infinite values in continuous x-axis
```

```
Warning: Transformation introduced infinite values in continuous y-axis
```



In ggplot, the scale_x_continuous (trans = "log2") and scale_y_continuous (trans = "log2") argument can be used to make plot both axes on a log scale.

Log transformations are super useful when our data is skewed and measured over a wide range like this. We can use different log transformations like base10 or natural logs but we most often prefer log2 units.

```
# Treated / Control
log2(10/10)
```

```
[1] 0
```

What if there was a doubling

```
#Treated/control
log2(10/20)
```

```
[1] -1
```

```
#Treated/control
log2(20/10)
```

```
[1] 1
```

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

```
[1] 2
```

Let's add a log2 fold-change column to our little 'meancounts' data.frame:

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

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

NaN is not a number while Inf is infinity (in this case -Inf is negative infinity). When NaN is returned, the number is divided by zero before the log is taken. When infinity is returned, the log of zero was taken. (genes with zero expression).

We need to remove the zeros

```
#Find the zeroes
zero.vals <- which(meancounts[,1:2]==0, arr.ind=TRUE)
to.rm <- unique(zero.vals[,1])
#Remove the zeroes
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
```

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

```
                control.mean treated.mean log2fc
ENSG00000000005         0.00         0.00    NaN
ENSG00000000938         0.75         0.00   -Inf
ENSG00000004848         0.00         0.25    Inf
ENSG00000004948         0.00         0.00    NaN
ENSG00000005001         0.00         0.00    NaN
ENSG00000005102         1.00         0.00   -Inf
```

The above code gives us the OPPOSITE of what I want. Adding an ! will flip trues and falses

```
to.rm.inds <- rowSums(meancounts[,1:2] == 0) >0
mycounts <- meancounts [!to.rm.inds,]
dim(mycounts)
```

```
[1] 21817     3
```

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 helps ensure that the exact position of the TRUE values are returned– which geners and samples have zeroes. The

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

A common threshold used for calling something deferentially expressed is a log2(FoldChange) of greater than 2 or less than -2.

let's filter the dataset both ways to see how many genes are up or down-regulated

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

```
sum(up.ind)
```

```
[1] 250
```

There are 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?

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

```
[1] 367
```

There are 367 down-regulated genes

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

We haven't done any statistical analysis to figure out if these differences are actually significant; we can tell that there may be relatively large differences between genes, but we can't say for sure if the differences are statistically significant. The differences could be due to natural variance of the gene (nothing to do with the treatment!).

Setting up for DESeq

```
 library(DESeq2)
citation("DESeq2")
```

```
To cite package 'DESeq2' in publications use:

  Love, M.I., Huber, W., Anders, S. Moderated estimation of fold change
  and dispersion for RNA-seq data with DESeq2 Genome Biology 15(12):550
  (2014)
```

```
A BibTeX entry for LaTeX users is

  @Article{,
    title = {Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2
    author = {Michael I. Love and Wolfgang Huber and Simon Anders},
    year = {2014},
    journal = {Genome Biology},
    doi = {10.1186/s13059-014-0550-8},
    volume = {15},
    issue = {12},
    pages = {550},
  }
```

We will use the 'DESeqDataSetFromMatrix' function to build the required object (called dds).

```
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
```

Get our results back from the 'dds' object

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

```
estimating size factors

estimating dispersions

gene-wise dispersion estimates

mean-dispersion relationship

final dispersion estimates

fitting model and testing
```

Get our results back from the 'dds' object

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

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 6 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
                 <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
                     padj
                <numeric>
ENSG00000000003  0.163035
ENSG00000000005        NA
ENSG00000000419  0.176032
ENSG00000000457  0.961694
ENSG00000000460  0.815849
ENSG00000000938        NA
```

```
summary(res)
```

```
out of 25258 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)       : 1563, 6.2%
LFC < 0 (down)     : 1188, 4.7%
outliers [1]       : 142, 0.56%
low counts [2]     : 9971, 39%
(mean count < 10)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

The default alpha value is 0.1 so the adjusted p value cutoff will be a number less than 0.1. We can manipulate that by changing the alpha value.

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

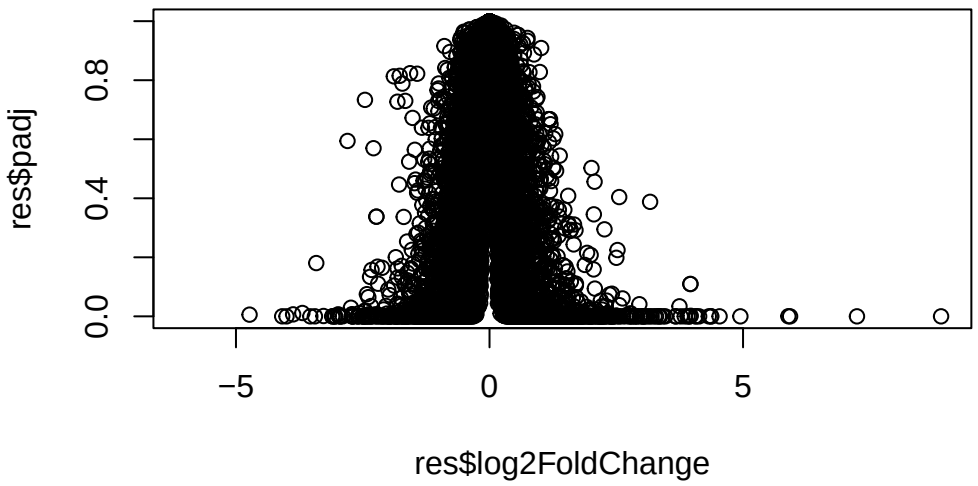
```
out of 25258 with nonzero total read count
adjusted p-value < 0.05
LFC > 0 (up)       : 1236, 4.9%
LFC < 0 (down)     : 933, 3.7%
outliers [1]       : 142, 0.56%
low counts [2]     : 9033, 36%
(mean count < 6)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results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

A Summary Results Plot

Volcano plot.

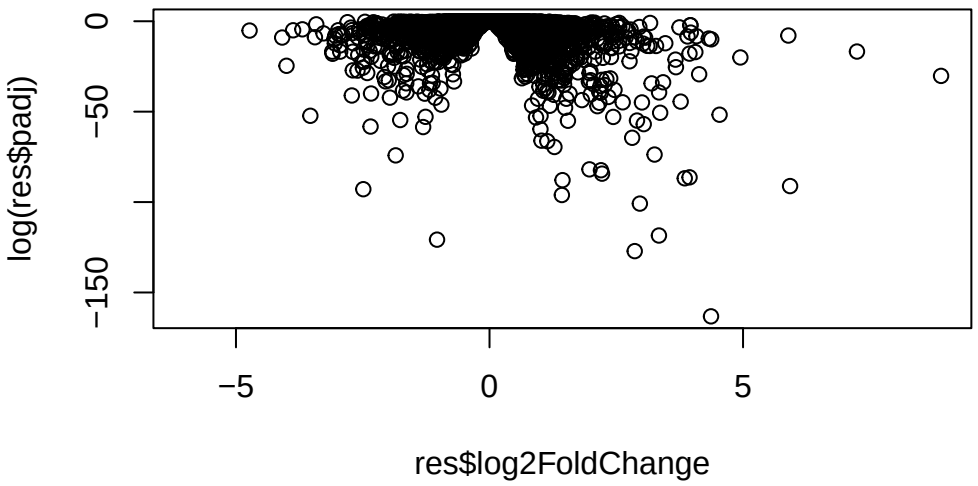
This is a common type of summary figure that keeps both out inner biologist and inner stats nerd happy because is shows both p-values and log2(FoldChanges)

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



Taking the log

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




```
log(0.1)
```

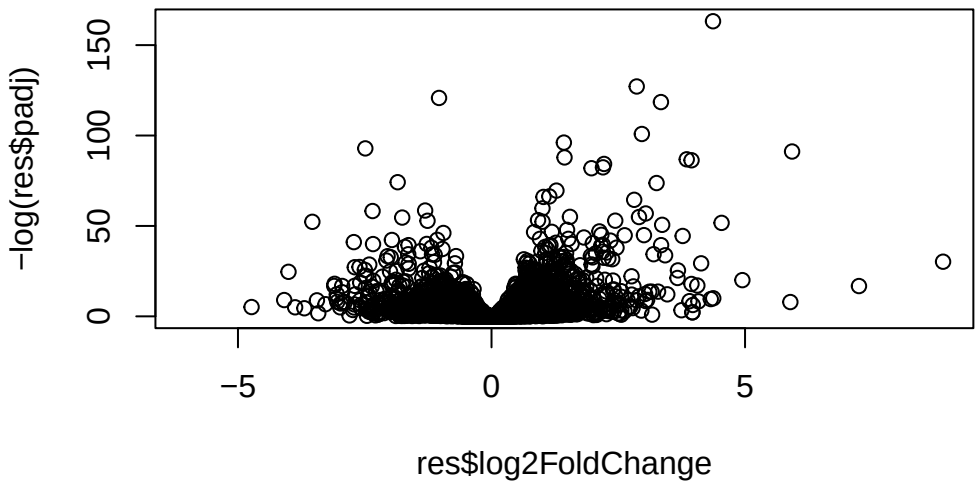
```
[1] -2.302585
```

```
log(0.001)
```

```
[1] -6.907755
```

Second value is more negative!

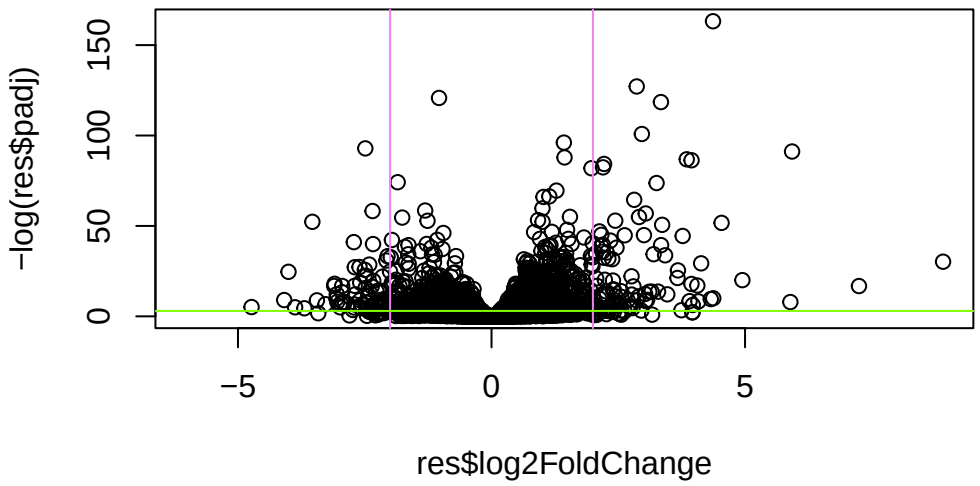
```
plot( res$log2FoldChange, -log(res$padj))
```



More visually pleasing to add negative to log argument.

Volcano plots highligh the proportion of genes that are both significantly regulated (large difference between treated and control) AND are statistically significant!

```
plot( res$log2FoldChange, -log(res$padj))
abline(v=2, col = "violet")
abline (v=-2, col ="violet")
abline (h=-log(0.05), col = "chartreuse")
```

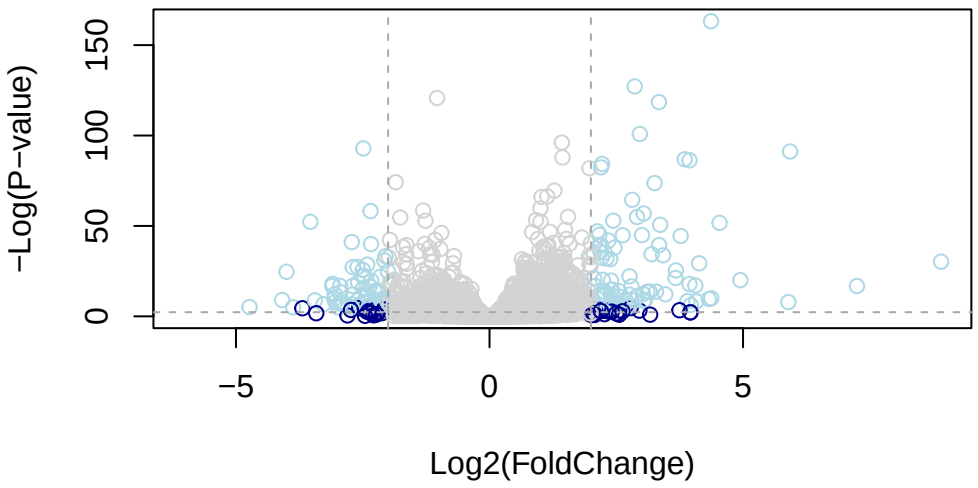



```
mycols <- rep("lightgray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ]  <- "darkblue"

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

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

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



Save our results to date

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

```
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
```

Adding Annotation Data

We need to add alternative gene names and extra annotation to our results table (only has Ensembl gene IDs right now)

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

```
Warning: package 'AnnotationDbi' was built under R version 4.3.2
```

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

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

```
 [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
 [6] "ENTREZID"     "ENZYME"       "EVIDENCE"     "EVIDENCEALL"  "GENENAME"
[11] "GENETYPE"     "GO"           "GOALL"        "IPI"          "MAP"
[16] "OMIM"         "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"
[21] "PMID"         "PROSITE"      "REFSEQ"       "SYMBOL"       "UCSCKG"
[26] "UNIPROT"
```

The main function we will use here is called 'mapIds()'

Out current IDs are here:

```
head(row.names(res))
```

```
[1] "ENSG00000000003" "ENSG00000000005" "ENSG00000000419" "ENSG00000000457"
[5] "ENSG00000000460" "ENSG00000000938"
```

These are in ENSEMBLE format. I want "SYMBOL" ids:

```
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       FIRRM
ENSG00000000938        NA         FGR
```

Let's add GENENAME

```
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 8 columns
                 baseMean log2FoldChange     lfcSE      stat    pvalue
                <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
                     padj      symbol               genename
                <numeric> <character>            <character>
ENSG00000000003  0.163035      TSPAN6          tetraspanin 6
ENSG00000000005        NA        TNMD            tenomodulin
ENSG00000000419  0.176032        DPM1 dolichyl-phosphate m..
ENSG00000000457  0.961694       SCYL3 SCY1 like pseudokina..
ENSG00000000460  0.815849       FIRRM FIGNL1 interacting r..
ENSG00000000938        NA         FGR FGR proto-oncogene, ..
```

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),
                      #column="UNIPROT",
                      #keytype="ENSEMBL",
                      #multiVals="first")
```

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

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 9 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
                 <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
                     padj      symbol               genename      entrez
                <numeric> <character>            <character> <character>
ENSG00000000003  0.163035      TSPAN6          tetraspanin 6        7105
ENSG00000000005        NA        TNMD            tenomodulin       64102
ENSG00000000419  0.176032        DPM1 dolichyl-phosphate m..        8813
ENSG00000000457  0.961694       SCYL3 SCY1 like pseudokina..       57147
ENSG00000000460  0.815849       FIRRM FIGNL1 interacting r..       55732
ENSG00000000938        NA         FGR FGR proto-oncogene, ..        2268
```

Pathway Analysis

We will use the **gage** package along with **pathview** here to do gene set enrichment (a.k.a pathway analysis) and figure generation respectively

```
library(pathview)
library(gage)
library(gageData)
```

```
# examine the first two pathways in the kegg set for humans
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"
```

What we need for 'gage()' is out genes in ENTREZ id format with a measure of their importance.

It wants a vector of e.g. fold-changes.

```
foldchanges <- res$log2FoldChange
head(foldchanges)
```

```
[1] -0.35070302          NA  0.20610777  0.02452695 -0.14714205 -1.73228897
```

```
x <- c(100, 80, 100)
names(x) <- c("desteny", "barry", "chris")
x
```

```
desteny   barry   chris
    100      80     100
```

Add ENTREZ ids as 'names()' to my 'foldchanges' vector.

```
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 can run a **gage** pathway analysis with this input vector and the geneset we want to examine for overlap/enrichment

```
keggres = gage(foldchanges, gsets=kegg.sets.hs)
#Look at these results
attributes (keggres)
```

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

```
head(keggres$less, 3)
```

```
                                      p.geomean stat.mean        p.val
hsa05332 Graft-versus-host disease 0.0004250461 -3.473346 0.0004250461
hsa04940 Type I diabetes mellitus  0.0017820293 -3.002352 0.0017820293
hsa05310 Asthma                    0.0020045888 -3.009050 0.0020045888
                                         q.val set.size         exp1
hsa05332 Graft-versus-host disease 0.09053483       40 0.0004250461
hsa04940 Type I diabetes mellitus  0.14232581       42 0.0017820293
hsa05310 Asthma                    0.14232581       29 0.0020045888
```

We can view these pathways with our geneset genes higlighted using the 'pathview()' function. E.g. for Asthma I will use the pathway.id hsa05310 as seen above.

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

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

```
Info: Working in directory /Users/adithiii/BIMM 143 R /Class 13
```

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

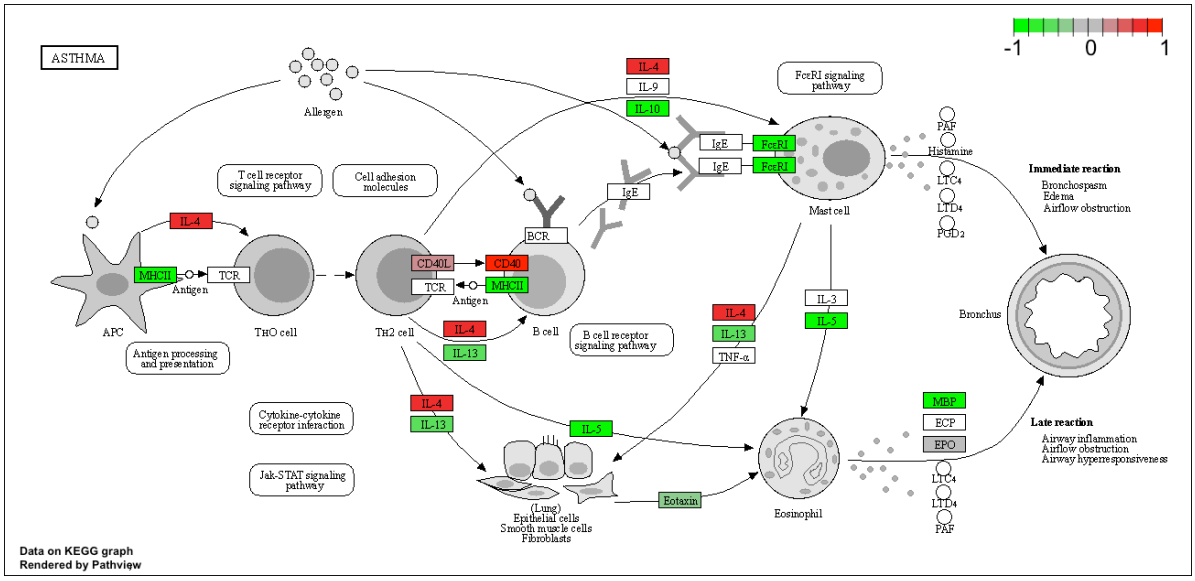


Figure 1: My genes involved in Asthma pathway