Class 13 transcriptomics

Ani A16647613

The puprose of this lab is to look at differential expression analysis.

First, set up Bioconductor

Within our console, we will install BiocManager, and within it, we will insall DESeq2 bioconductor.

Importing countData and colData

We will now take a look at the countdata to see the number of readings from each gene per sample and metadata (coldata) to see the columns of the count data

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

A look at the counts:

```
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
```

A look at the metadata:

```
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
```

Checking correspondence of count and metadata:

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

```
[1] TRUE
```

Q1. How many genes are in this dataset?

There are 38694 genes in this dataset.

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

There are 4 control cell lines in the dataset.

Toy differential gene expression

Now, we are going to extract and summarize control samples, to compare control vs treated.

To do so, we need to use metadata (dex):

```
control <- metadata[metadata[,"dex"]=="control",]
control.counts <- counts[ ,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
ENSG00000000938
           0.75
```

Here is also the alternative way:

```
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
ENSG00000000938
           0.75
```

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

Rather than using `rowSums` and dividing the `control.counts` by 4, you can use `rowMeans(control.counts)` to make the code more robust. Let's try it out:

```
control <- metadata %>% filter(dex=="control")
control.counts <- counts %>% select(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
```

Q4. Follow the same procedure for the `treated` samples (e.g. 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 <- rowMeans(treated.counts)
head(treated.mean)
```

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

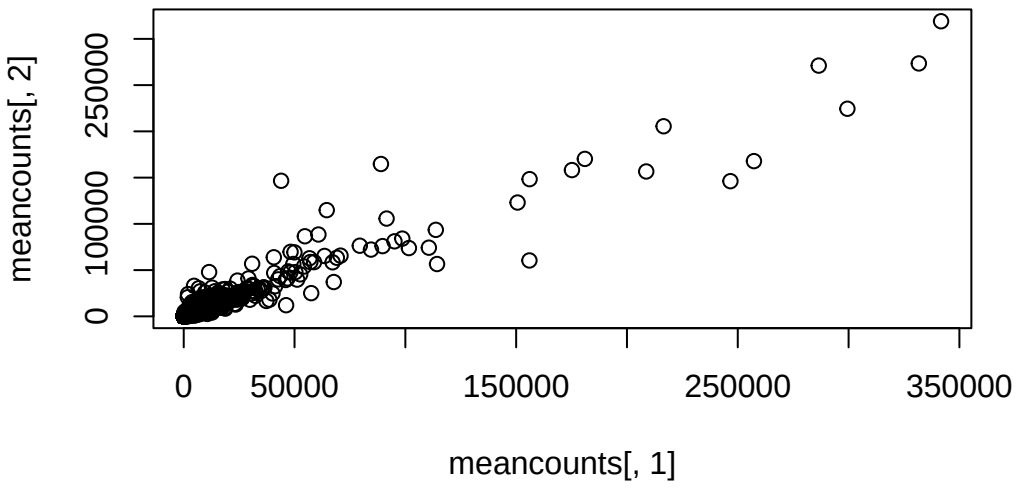
To save the results of the means in a new dataframe, we will create a new data frame titled `meancounts`

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

. Q5A. Creating a scatterplot showing the mean of treated samples against the mean of the control samples.

Now to plot and analyze the dataframe means:

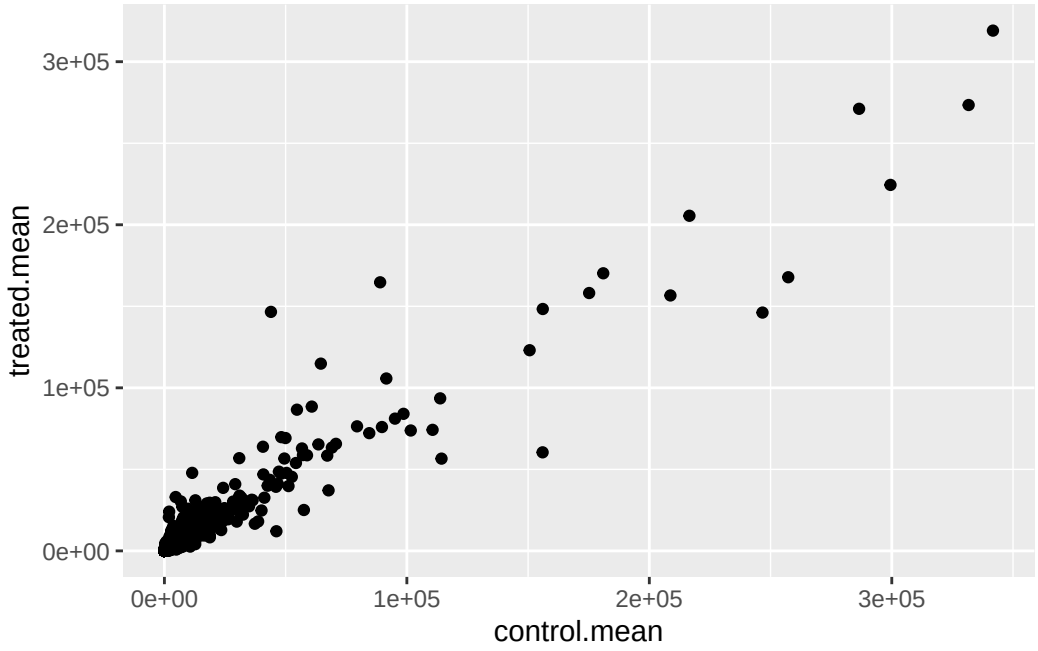
```
plot(meancounts[,1], meancounts[,2])
```



. Q5B. Using ggplot package, make the figure shown below and you would use the `geom_point()` function, since we are doing a scatterplot.

Now as a ggplot to make the dataset more succinct and "pleasing" appearance-wise:

```
library(ggplot2)

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



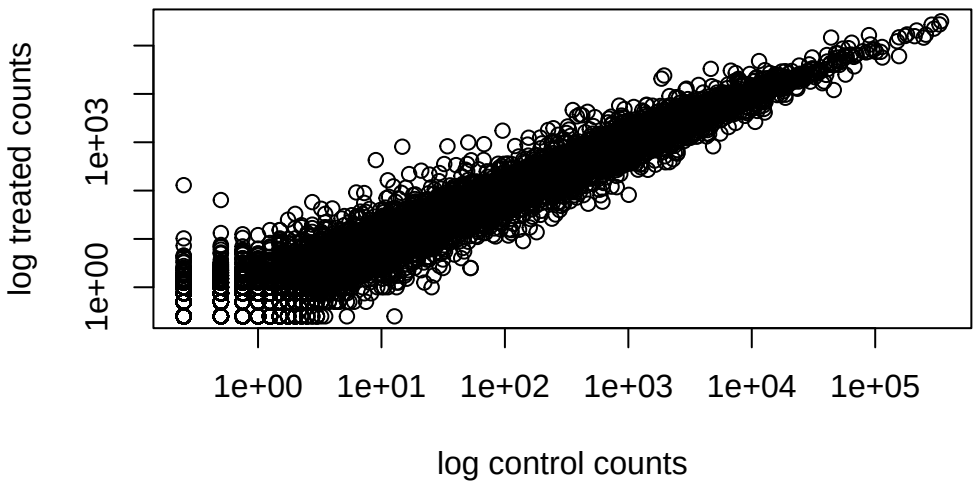
Majority of the points in the dataset are skewed to the left, as shown in the ggplot and regular plot. Therefore, we would need to use a log-log plot to properly analysed these datas.

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

```
plot(meancounts[,1], meancounts[,2], log="xy", xlab="log control counts", ylab= "log treat
```

```
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
```



Now to apply this to log2 transformations

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

```
[1] 0
```

Indicates that there is no change in ID.

Note: This log2 transformation has this nice property as mentioned above and if the value is doubles, the value will be 1. If the value is halved it will be -1.

Now we can add the log2 fold change to our current results

```
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
```

Negative log2fc values indicates that the mean value went down upon treatment implementation compared to control. "NaN" and "-Inf" is inconclusive and no change, so we can remove it from the dataset.

To get rid of the inconclusive/unwanted results, we can use this function:

```
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?

arr.ind in the which() function to indicate which rows and columns the "TRUE" values are found in (more specifically in this case the "rows")

we would need to call the "unique()" function to signify which genes to remove since they are the zero genes.

```
nrow(mycounts)
```

```
[1] 21817
```

Above shows how many genes are left.

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

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

```
[1] 250
```

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

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

```
[1] 367
```

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

We find the quantification of the changes useful however, we would need to find the statistical significance of the changes in order to fully trust our computations. We would need to determine is the changes in the data are significant.

Working with DESeq Analysis

DEseq2 package was installed at the beginning of the lab but now we are going to add it to our library.

```
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, 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 objects are masked from 'package:dplyr':

    first, rename

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

    findMatches

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

    expand.grid, I, unname

Loading required package: IRanges


Attaching package: 'IRanges'

The following 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, 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
```

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

```
estimating size factors

estimating dispersions

gene-wise dispersion estimates

mean-dispersion relationship

final dispersion estimates

fitting model and testing
```

```
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>
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
```

Now to get some basic summary tallies with the `summary` function (not extremely useful)

```
summary(res, alpha=0.05)
```

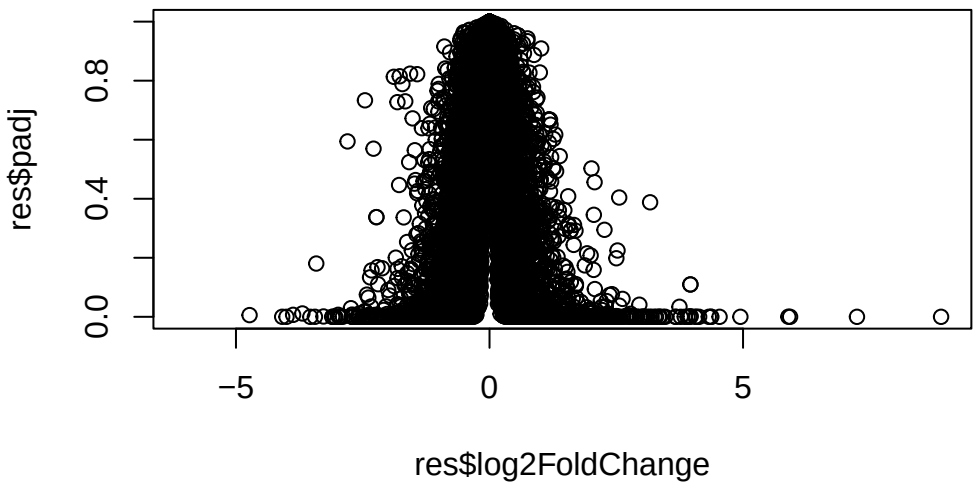
```


out of 25258 with nonzero total read count
adjusted p-value < 0.05
```

```
LFC > 0 (up)       : 1242, 4.9%
LFC < 0 (down)     : 939, 3.7%
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
```

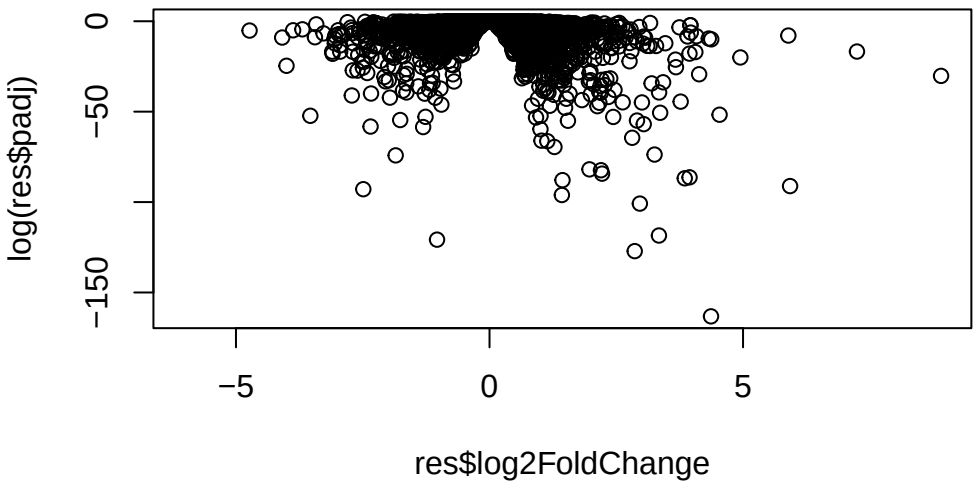
Volcano plot

To have a better analysis of the results, we can make a summary plot:

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

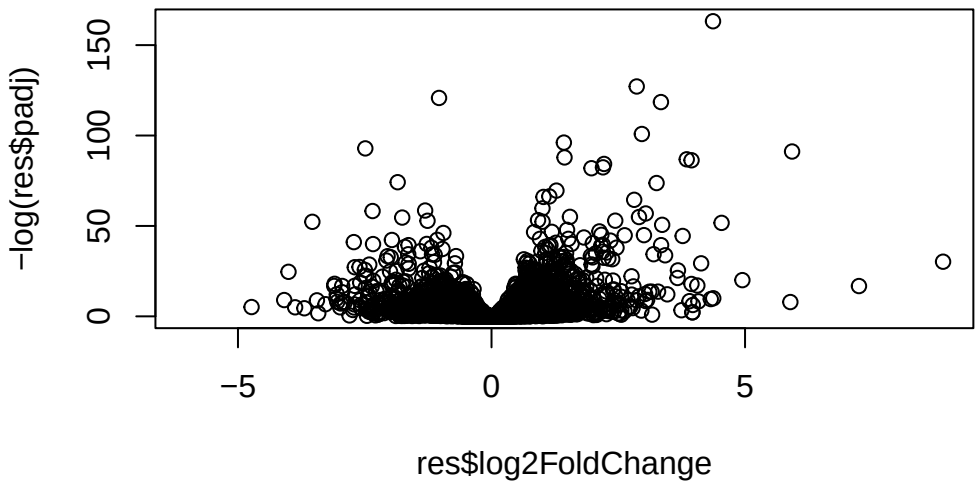



```
plot(res$log2FoldChange, log(res$padj))
```



We would care more about the values on the bottom, which are still quite hard to read, so we should adjust the `log(res$padj)` values:

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



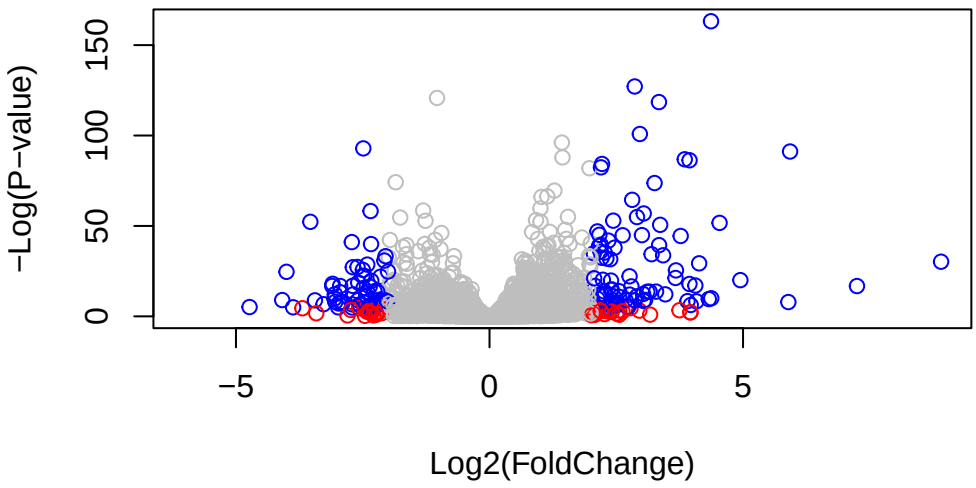
This is the volcano plot, and we would focus on the "erupting" points, which are the genes that change significantly in larger quantities.

For better visualization, we can add custom color vectors to indicate transcripts that contain the large significant changes in conditions by incorporating it into the following codes:

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

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

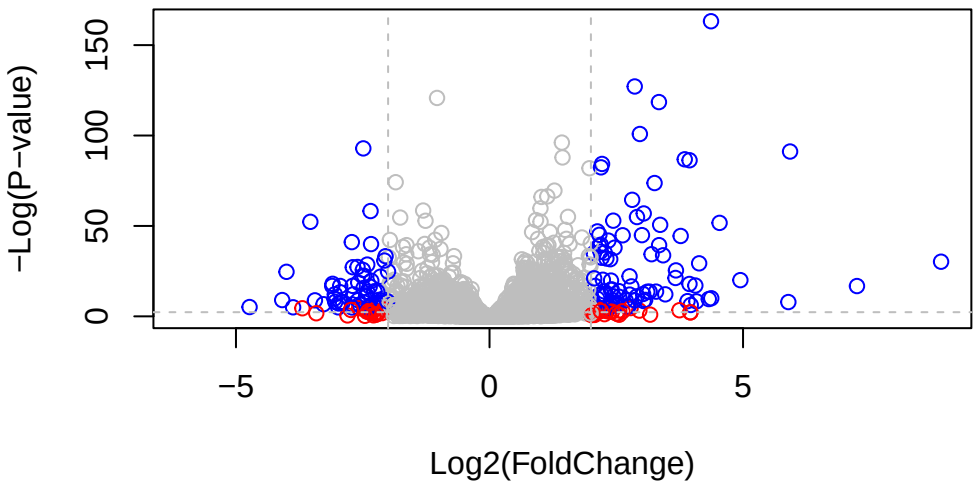
Now to incorporate it into the volcano plot:

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



To establish the limit line to signify the cut-off marks, we can add lines:

```
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)
```



From Class 14 : Add Annotation Data

```
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
```

Downloaded AnnotationDbi and org.Hs.eg.db

```
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] "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"
```

```
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)
```

```
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
```

We also want to identify entrez (IDs)

```
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
```

```
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      entrez
                <numeric> <character> <character>
ENSG00000000003  0.163035      TSPAN6        7105
ENSG00000000005        NA        TNMD       64102
ENSG00000000419  0.176032        DPM1        8813
ENSG00000000457  0.961694       SCYL3       57147
ENSG00000000460  0.815849       FIRRM       55732
ENSG00000000938        NA         FGR        2268
```

Now for Genename IDs;

```
res$name <- 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 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      entrez                   name
                <numeric> <character> <character>            <character>
ENSG00000000003  0.163035      TSPAN6        7105          tetraspanin 6
ENSG00000000005        NA        TNMD       64102            tenomodulin
ENSG00000000419  0.176032        DPM1        8813 dolichyl-phosphate m..
```

```
ENSG00000000457  0.961694       SCYL3       57147 SCY1 like pseudokina..
ENSG00000000460  0.815849       FIRRM       55732 FIGNL1 interacting r..
ENSG00000000938        NA         FGR        2268 FGR proto-oncogene, ..
```

Pathway Analysis

Now that the essential annotation data were added, we can talk to different databases that use these IDs.

```
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)
```

Using the **gage** package we can do geneset analysis (also known as pathway analysis, genset enrichment, and overlap analysis)

Now, using KEGG

```
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 needs a named vector of fold changes, where the names of the values are Entrez gene IDs.

```
foldchange <- res$log2FoldChange
names(foldchange) <- res$entrez
head(foldchange)
```

```
      7105       64102        8813       57147       55732        2268
-0.35070302          NA  0.20610777  0.02452695 -0.14714205 -1.73228897
```

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

Let's look at what is in our results here

```
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
```

can use the return pathway IDs (hsa...) from KEGG as input to the `pathview` pathway to make figures with the highlighted DEGs

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

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

```
Info: Working in directory C:/Users/anban/OneDrive/Desktop/R lab/Class 13
```

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