Lab 13: RNA Seq Analysis

Marriane Allahwerdi (A16902759)

```
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
```

Q1.How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

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

4

```
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
```

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

```
[1] 4
```

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

```
control treated
      4       4
```

Toy differential expression analysis

calculate the mean per gene count values for all "control" samples (i.e columns in `counts`) and do the same for "treated" and then compare them.

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

1. Find all "control" valuesc/columns in `counts`

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

2. Find the mean per gene across all control columns.

```
control.mean <- apply(control.counts, 1, mean)
```

> 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)

3. Do the same steps to find the `treated.mean` values

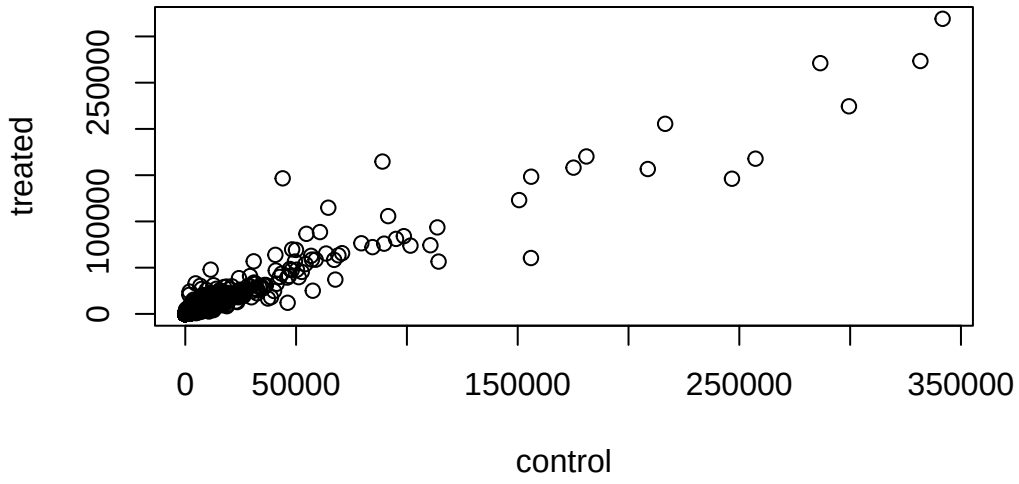
```
treated.inds <- metadata$dex == "treated"
treated.counts <- counts[,treated.inds]
```

```
treated.mean <- apply(treated.counts, 1, mean)
```

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.

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

```
plot(meancounts[,1], meancounts[,2], xlab="control", ylab="treated")
```

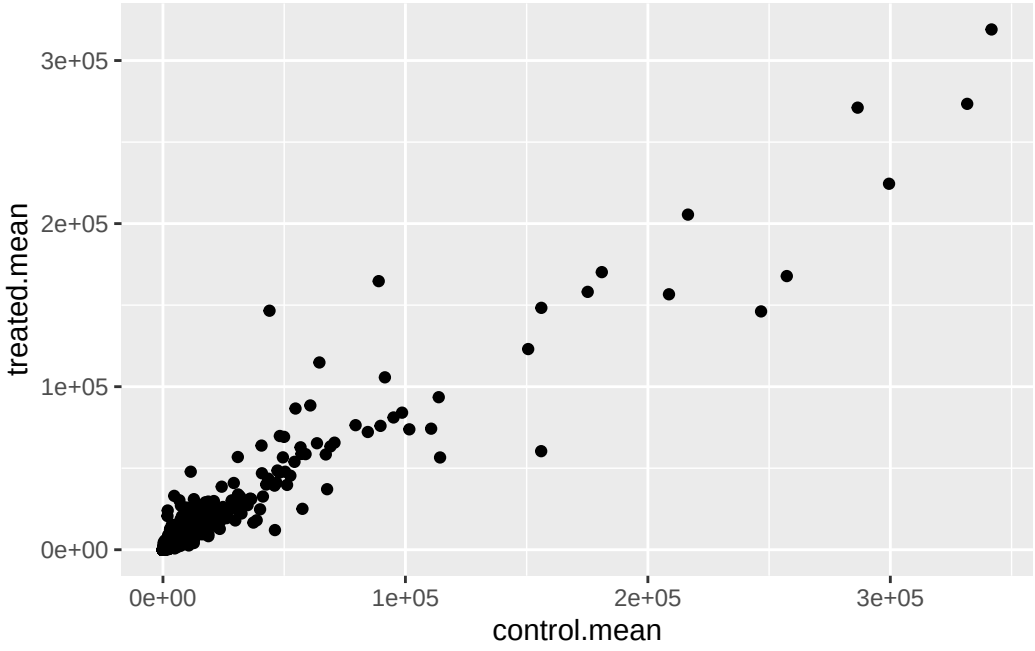



```
treated.mean <- apply(counts[, metadata$dex =="treated"], 2, mean)
```

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?

geom_point

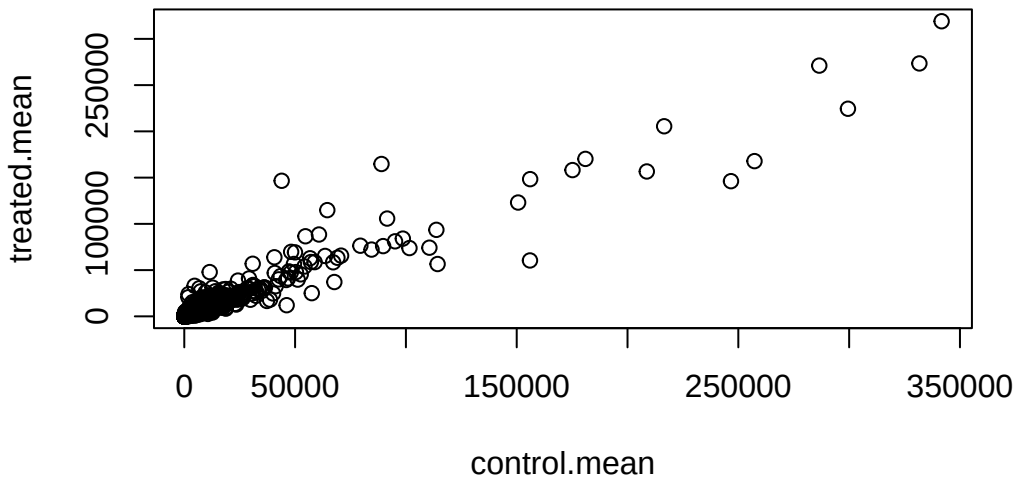
```
library(ggplot2)
ggplot(meancounts) + aes(x=control.mean, y=treated.mean) + geom_point()
```



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

log2

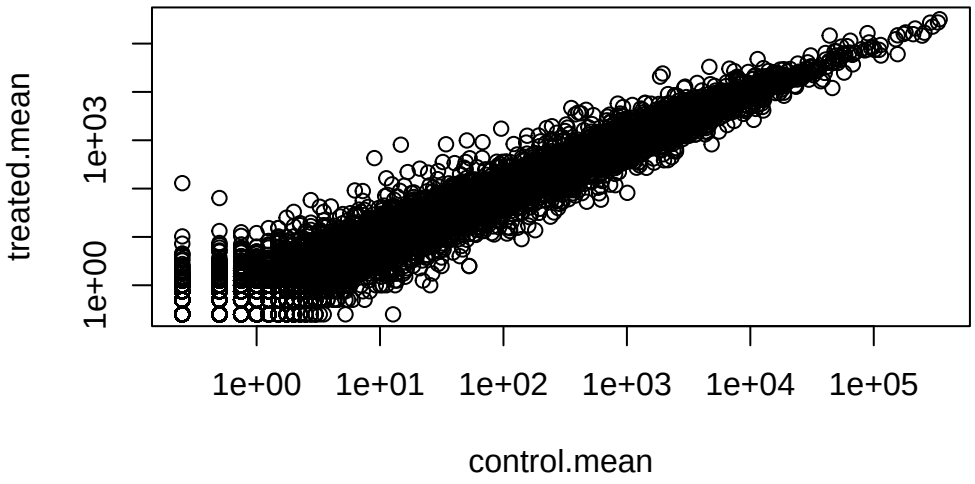
```
plot(meancounts)
```




```
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
```



We frequently use log2 transformations for this type of data.

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

```
[1] 0
```

```
log2(20/10)
```

```
[1] 1
```

```
log2(10/20)
```

```
[1] -1
```

These log2 values make interpretation of "fold-change" a little easier and a rule-of-thumb in the file is a log2 fold-change of +2 or -2 is where we start to pay attention.

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

```
[1] 2
```

Let's calculate the log2(fold-change) and add it to our **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
```

> 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=TRUE argument will cause which() to return both the row and column indices (i.e. positions) where there are TRUE values. In this case this will tell us which genes (rows) and samples (columns) have zero counts. We are going to ignore any genes that have zero counts in any sample so we just focus on the row answer

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

> Q. How many genes do I have left after this zero count filteriing?

```
nrow(mycounts)
```

```
[1] 21817
```

> Q8. How many genes are "up" regulated upon drug treatment, at threshold of +2 log2fc?

1. I need to extract the log2fc values
2. I need to find those that are above +2
3. Count them

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

```
[1] 250
```

Q9. How many genes are "down" regulated upon drug treatment, at threshold of +2 log2fc?

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

```
[1] 367
```

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

Wow hold on we are missing the stats here. So no we don't trust these results.

Is the difference in the mean counts significant??

Let's do this analysis the right way with stats and use `DESeq2` package

```
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, saveRDS, setdiff,
    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 first function that we will use will setup the data in the way (format) DESeq wants it.

```
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
```

The function in the package is called `DESeq()` and we can run it on our `dds` object

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

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

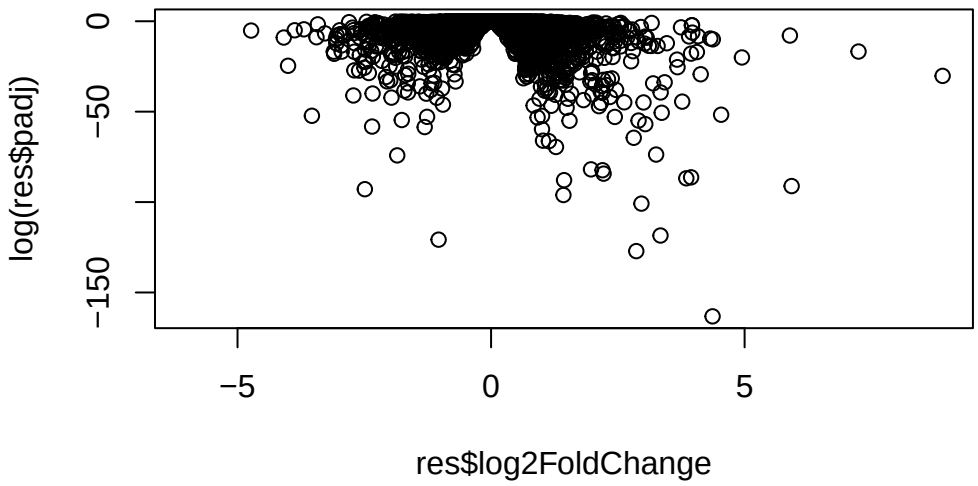
I will get the results from the dds with the `results` function:

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

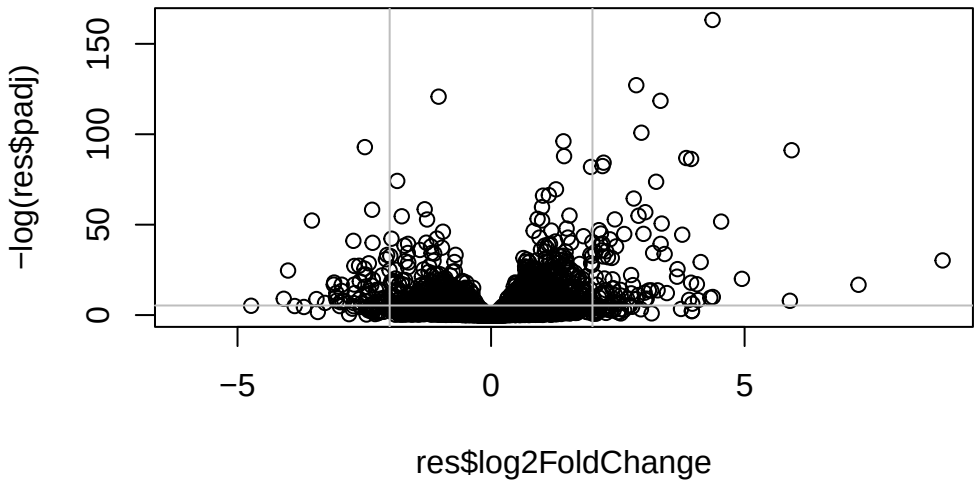
```
class: DESeqDataSet
dim: 6 8
metadata(1): version
assays(4): counts mu H cooks
rownames(6): ENSG00000000003 ENSG00000000005 ... ENSG00000000460
  ENSG00000000938
rowData names(22): baseMean baseVar ... deviance maxCooks
colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
colData names(5): id dex celltype geo_id sizeFactor
```

Make a common overall results figure from this analysis. This is designed to keep our inner biologist and inner stats nerd happy.

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

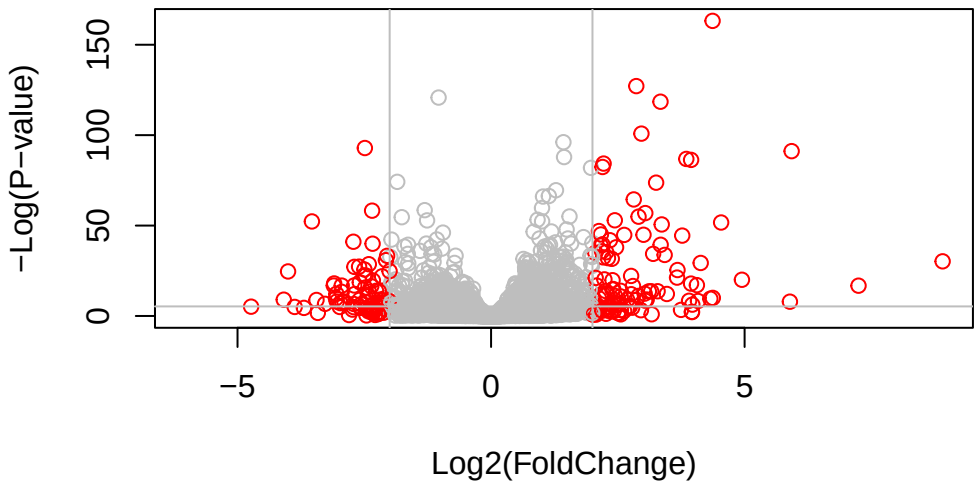



```
plot(res$log2FoldChange, -log(res$padj))
abline(v=c(-2,2), col="gray")
abline(h=-log(0.005), col="gray")
```



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


plot( res$log2FoldChange,  -log(res$padj),
 col=mycols, ylab="-Log(P-value)", xlab="Log2(FoldChange)" )
abline(v=c(-2,2), col="gray")
abline(h=-log(0.005), col="gray")
```



I want to save my results to date out to disc

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

We will pick up next day and add annotation (what are the genes of interest) and do pathway analysis (what biology) are they known to be involved with.

I need to translate our gene identifiers "ENSG000" into gene names that the rest of the world can understand.

Annotation

To do this "annotation" I will use "AnnotationDBi"

```
library(AnnotationDbi)
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"
```

I will use the `mapIds()` function to "map" my identifiers to those from different databases. I will go between "ENSEMBL" and "SYMBOL" (and then after "GENENAME").

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

```
'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
```

Add "GENENAME"

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

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

Add "ENTREZID"

```
res$entrezid<- mapIds(org.Hs.eg.db,
              keys = rownames(res),
              keytype = "ENSEMBL",
              column = "ENTREZID")
```

```
'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               genename    entrezid
                <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
```

Save our annotated results object

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

Pathway Analysis

Now that we have our results with added annotation we can do some pathway mapping.

Let's use the **gage** package to look for KEGG pathways in our results (genes of interest). I will also use the **pathview** package to draw little pathway figures.

```
library(pathview)
```

```
#############################################################################
Pathview is an open source software package distributed under GNU General
Public License version 3 (GPLv3). Details of GPLv3 is available at
http://www.gnu.org/licenses/gpl-3.0.html. Particullary, users are required to
formally cite the original Pathview paper (not just mention it) in publications
or products. For details, do citation("pathview") within R.

The pathview downloads and uses KEGG data. Non-academic uses may require a KEGG
license agreement (details at http://www.kegg.jp/kegg/legal.html).
#############################################################################
```

```
library(gage)
```

```
library(gageData)

data(kegg.sets.hs)

# Examine the first pathway
head(kegg.sets.hs, 1)
```

```
$`hsa00232 Caffeine metabolism`
[1] "10"   "1544" "1548" "1549" "1553" "7498" "9"
```

What **gage** wants as input is not my big table /data.frame of results with everything in it. It just wants a "vector of importance". For RNASeq data like we have this is our log2FC values....

```
foldchanges = res$log2FoldChange
names(foldchanges) = res$entrez
head(foldchanges)
```

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

```
x <- c(10, 20, 100)
names(x) <- c("barry", "ana", "jk")
x
```

```
barry   ana    jk
   10    20   100
```

Now, let's run the gage pathway analysis.

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

What is this `keggres` object?

```
attributes(keggres)
```

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

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

```
                                      p.geomean stat.mean p.val q.val
hsa00232 Caffeine metabolism                 NA       NaN    NA    NA
hsa00983 Drug metabolism - other enzymes     NA       NaN    NA    NA
hsa01100 Metabolic pathways                  NA       NaN    NA    NA
                                      set.size exp1
hsa00232 Caffeine metabolism                 0   NA
hsa00983 Drug metabolism - other enzymes     0   NA
hsa01100 Metabolic pathways                  0   NA
```

Let's use the pathview package to look at one of these highlighted KEGG pathways with our genes highlighted. "hsa05310 Asthma"

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

```
Warning: None of the genes or compounds mapped to the pathway!
Argument gene.idtype or cpd.idtype may be wrong.
```

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

```
Info: Working in directory /Users/marriane/Desktop/Class 13
```

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

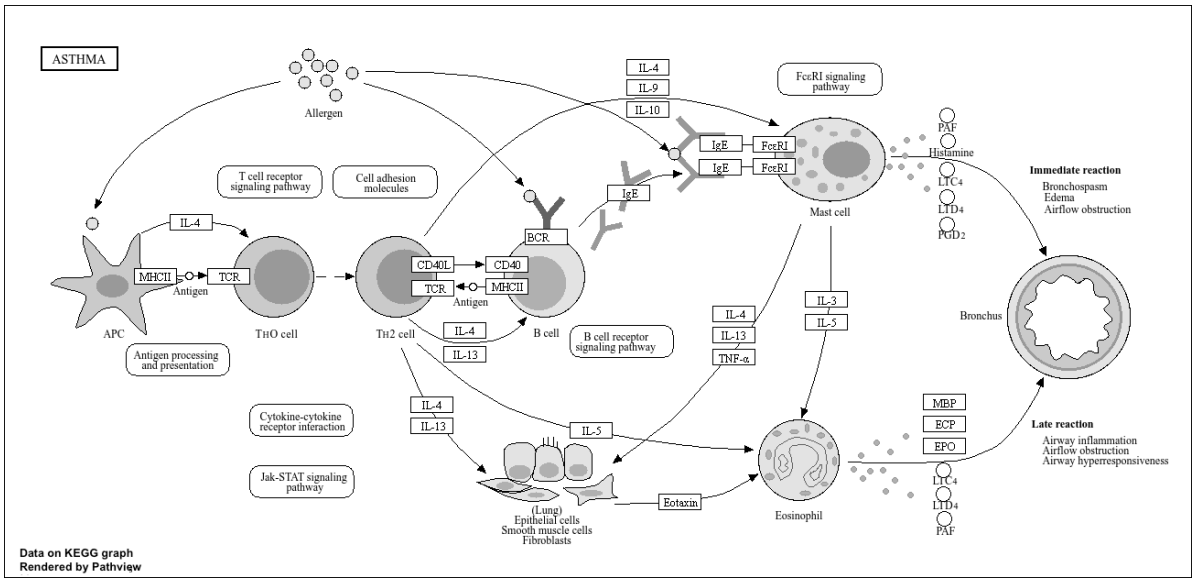


Figure 1: Asthma Pathway with my DEGs