class 12

Faisal

#Import 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
```

Q1 How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

Q2 How many ‘control’ cell lines do we have?

```
ncol(counts)
```

```
[1] 8
```

and the metadataaka "colData"

```
(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
7 SRR1039520 control  N061011 GSM1275874
8 SRR1039521 treated  N061011 GSM1275875
```

Lets make sure that the id column of the metadatamatch the order of the columns in CountData.

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

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

We can use the 'all() function to check that all its input are TRUE

```
all( c(T,T,T, F))
```

```
[1] FALSE
```

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

```
[1] TRUE
```

Analysis by hand

```
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
7 SRR1039520 control  N061011 GSM1275874
8 SRR1039521 treated  N061011 GSM1275875
```

Lets first extract our counts for control samples to compare this to the count for treated (i.e with drug) samples

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

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

I want a single summary counts value for each gene in the control experiments. I will start by taking the average

> 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)

```
##apply(control.counts, 1, mean)
treated.mean <- rowMeans(control.counts)
treated.inds <- metadata$dex == "treated"
treated.ids <- metadata$id[ control.inds]
treated.counts = counts[, treated.ids ]
head(treated.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
```

```
treated.mean = rowMeans(treated.counts)
```

Now we do the same for the treated samples Please :-)

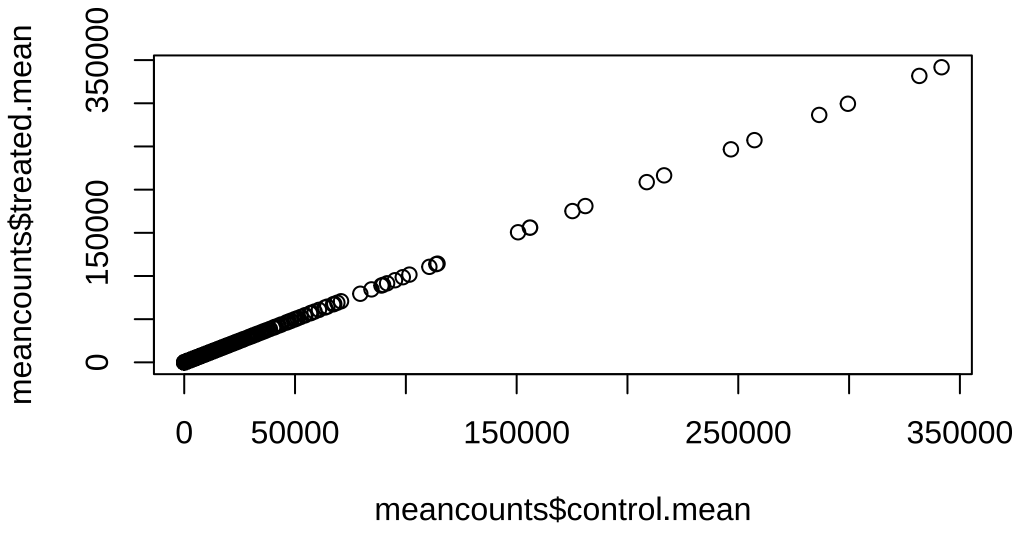
```
meancounts <- data.frame(control.mean, treated.mean)
head(meancounts)
```

```
                control.mean treated.mean
ENSG00000000003       900.75       900.75
ENSG00000000005         0.00         0.00
ENSG00000000419       520.50       520.50
ENSG00000000457       339.75       339.75
ENSG00000000460        97.25        97.25
ENSG00000000938         0.75         0.75
```

and make a wee plot to see how we are doing

> Q5 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$control.mean, meancounts$treated.mean)
```



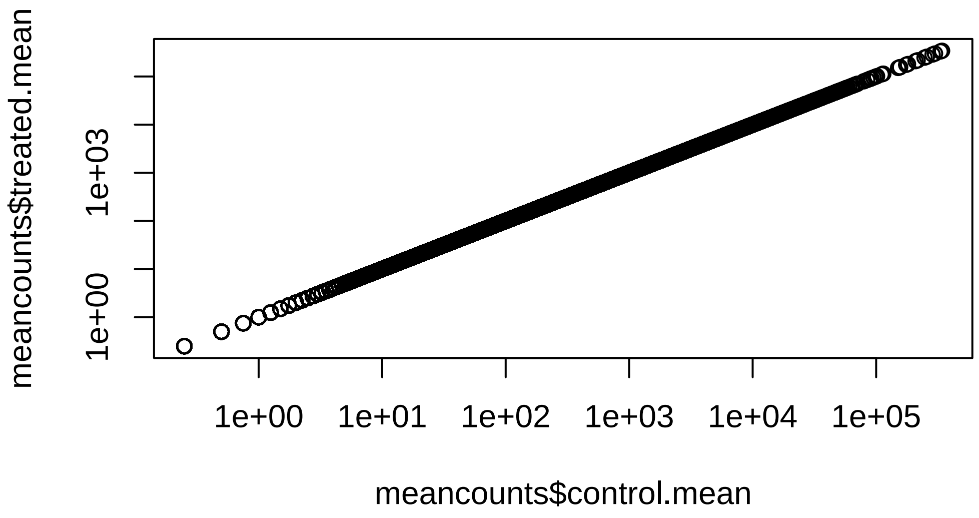
This screams for a log transformation so we can see our data

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

```r
plot(meancounts$control.mean, meancounts$treated.mean, 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): 15032 y values <= 0 omitted
from logarithmic plot
```



The most useful and most straightforward to understand is log2 transformation

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

```
[1] 0
```

Doubling

```r
log2(40/20)
```

```
[1] 1
```

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

```
[1] -1
```

add a "log2 fold-change"

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

```r
head(meancounts)
```

```
                control.mean treated.mean log2fc
ENSG00000000003       900.75       900.75      0
ENSG00000000005         0.00         0.00    NaN
ENSG00000000419       520.50       520.50      0
ENSG00000000457       339.75       339.75      0
ENSG00000000460        97.25        97.25      0
ENSG00000000938         0.75         0.75      0
```

Hmmm… we need to get rid of the genes where we have no count data as taking the log2 of these 0 counts does not tell us anything.

```r
head( meancounts == 0)
```

```
                control.mean treated.mean log2fc
ENSG00000000003        FALSE        FALSE   TRUE
ENSG00000000005         TRUE         TRUE     NA
ENSG00000000419        FALSE        FALSE   TRUE
ENSG00000000457        FALSE        FALSE   TRUE
ENSG00000000460        FALSE        FALSE   TRUE
ENSG00000000938        FALSE        FALSE   TRUE
```

```r
to.keep <- rowSums(meancounts[,1:2] == 0) == 0

mycounts <- meancounts[to.keep,]
head(mycounts)
```

```
                control.mean treated.mean log2fc
ENSG00000000003       900.75       900.75      0
ENSG00000000419       520.50       520.50      0
ENSG00000000457       339.75       339.75      0
ENSG00000000460        97.25        97.25      0
ENSG00000000938         0.75         0.75      0
ENSG00000000971      5219.00      5219.00      0
```

> 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?

it returns the true value.. we use the unique value because we don't need the position that has 2 trees, which is repeated.

How many genes are up regulated at the log2fc level of +2

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

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

```
[1] 0
```

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

and down regulated…

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

```
[1] 0
```

> Q10 No we dont trust these results because we dont know if the numbers are significant

We are missing the stats..

DESeq2 analysis

```r
library(DESeq2)
```

Like most bioconductor packages DESeq wants its input and output in a very specific format

```r
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
```

```r
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 main DESeq function is called DESeq

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

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

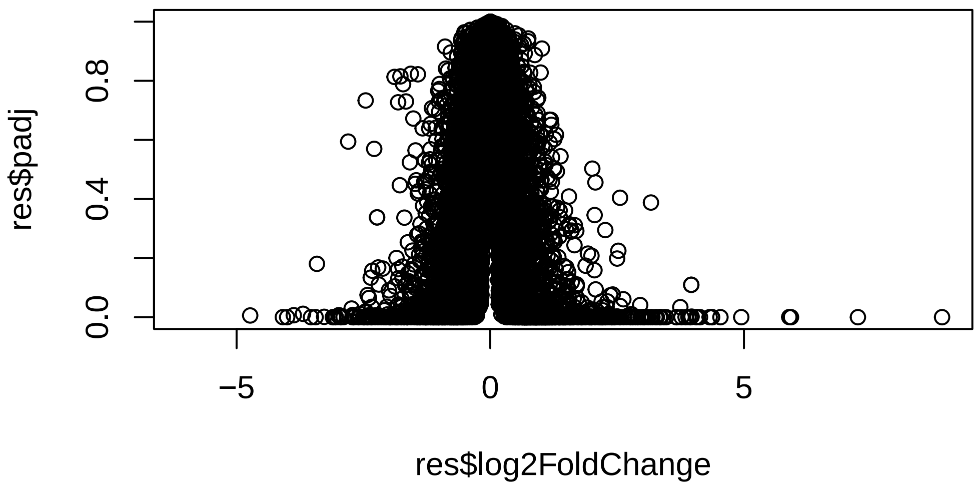
```r
res <- results(dds)
head(res)
```

```
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
```

Volcano plots

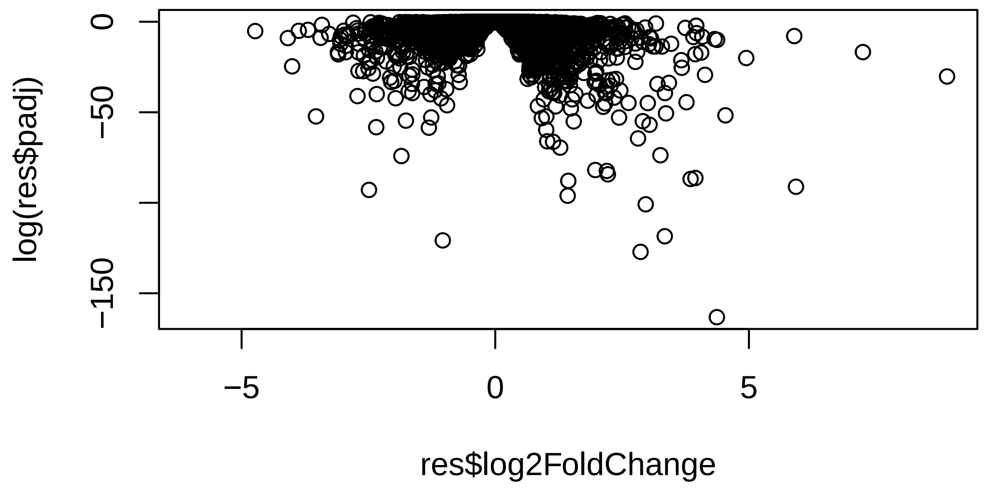
A major summary figure of this type of analysis is called a volcano plot - the idea here is to keep our inner biologist and inner stats person happy with one cool plot

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



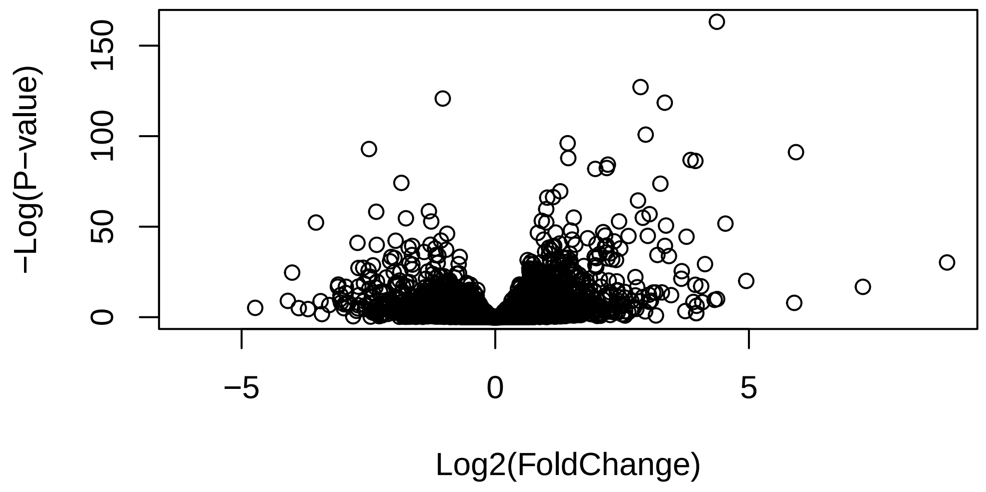
Improve this plot by taking the log of that p-value axis

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



I want to flip this y-axis so that the value i care about are at the top of the axis

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



gene annotation

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

```r
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"
```

#Pathway anaylyysis

```r
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).
##############################################################################
```

```r
library(gage)
```

```r
library(gageData)

data("kegg.sets.hs")

#examine the first 2 pathways in this keggg set for human

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"
```

```r
c(barry=4, clair=3, chandra=2)
```

```
  barry   clair chandra 
      4       3       2 
```

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

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

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

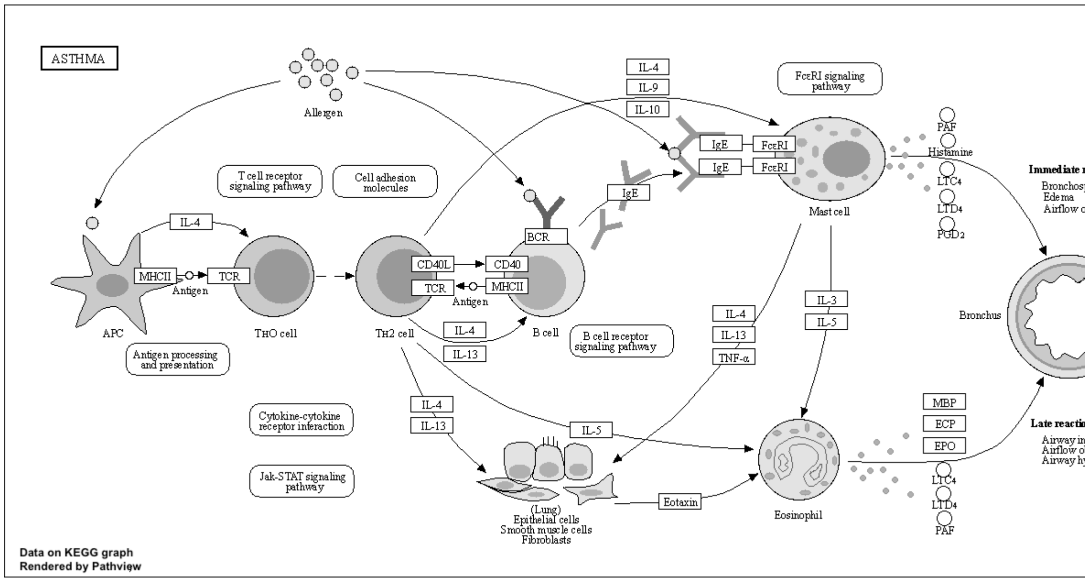
```
pathview(gene.data=foldchanges, pathway.id="hsa05310")
```

```
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/faisal/Desktop/UCSD/BIMM 143/Class 15
```

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

I put this in the document 
ASTHMA
Allergen
IL-4
IL-9
IL-10
FcεRI signaling pathway
IgE
FcεRI
PAF
Histamine
LTC4
LTD4
PGD2
Mast cell
T cell receptor signaling pathway
Cell adhesion molecules
IL-4
BCR
MHCII
Antigen
TCR
CD40L
CD40
APC
THO cell
TH2 cell
B cell
Antigen processing and presentation
B cell receptor signaling pathway
IL-13
IL-3
IL-5
TNF-α
Bronchus
Cytokine-cytokine receptor interaction
Jak-STAT signaling pathway
MBP
ECP
EPO
(Lung) Epithelial cells Smooth muscle cells Fibroblasts
Eotaxin
Eosinophil
Data on KEGG graph
Rendered by Pathview