Class 13: Transcriptomics

Sophia Wang (A16838155)

Today we will analyze some RNASeq data from Himes et al. on the effects of dexamethasone (dex), a synthetic glucocorticoid steroid on airway smooth muscle cells (ASM).

#Data import

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

```
                SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
ENSG00000000003        723        486        904        445       1170
ENSG00000000005          0          0          0          0          0
ENSG00000000419        467        523        616        371        582
ENSG00000000457        347        258        364        237        318
ENSG00000000460         96         81         73         66        118
ENSG00000000938          0          0          1          0          2
                SRR1039517 SRR1039520 SRR1039521
ENSG00000000003       1097        806        604
ENSG00000000005          0          0          0
ENSG00000000419        781        417        509
ENSG00000000457        447        330        324
ENSG00000000460         94        102         74
ENSG00000000938          0          0          0
```

```
head(metadata)
```

```
          id     dex celltype     geo_id
1 SRR1039508 control   N61311 GSM1275862
2 SRR1039509 treated   N61311 GSM1275863
3 SRR1039512 control  N052611 GSM1275866
4 SRR1039513 treated  N052611 GSM1275867
```

```
5 SRR1039516 control  N080611 GSM1275870
6 SRR1039517 treated  N080611 GSM1275871
```

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

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

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

```
[1] 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.

1. Find all "control" values/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)
```

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

Yes. apply() and 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)

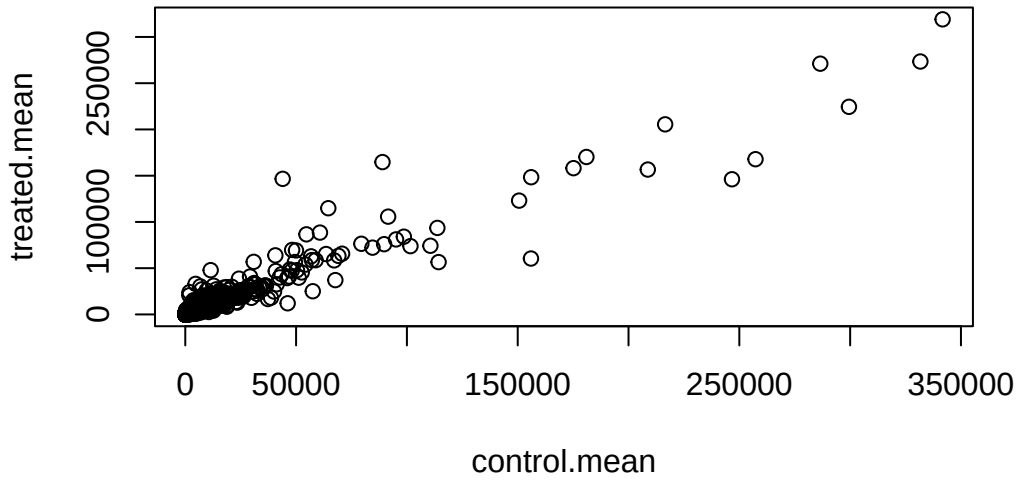
3. Do the same steps to find the `treated.mean`

```
treated.inds <- metadata$dex=="treated"
treated.counts <- counts[,treated.inds]
treated.mean <- apply(treated.counts,1,mean)
```

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

Q5 (a). Create a scatter plot showing the mean of the treated samples against the mean of the control samples.

```
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?

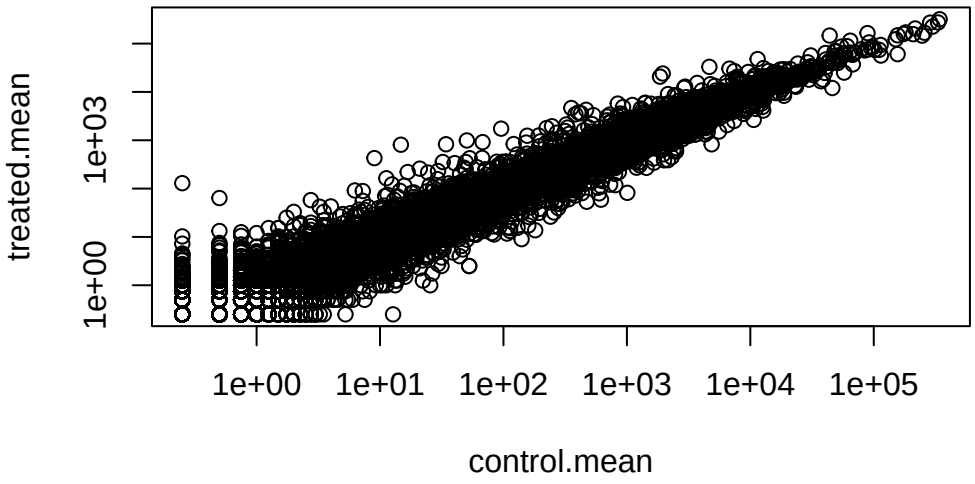
geom_point()

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

```
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 most 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 the interpretation of "fold-change" a little easier and a rule-of-thumb in the filed 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
```

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

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

```
nrow(my.counts)
```

```
[1] 21817
```

> Q7. What is the purpose of the arr.ind argument in the which() function call above? Why would we then take the first column of the output and need to call the unique() function?

arr.ind=TRUE argument will return both the row and column indices where there are TRUE values. Calling unique() will ensure we don't count any row twice if it has zero entries in both samples.

> Q8.How many genes are "up" regulated upon drug treatment with a threshold of +2 log2-fold-change?

```
sum(my.counts$log2fc>2)
```

```
[1] 250
```

> Q9.How many genes are "down" regulated upon drug treatment with a threshold of -2 log2-fold-change?

```
sum(my.counts$log2fc < -2)
```

```
[1] 367
```

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

No, missing significance and statistics. Is the difference in the mean counts significant?

Lets do this analysisi the right way with stats and use the **DESeq2** package

##DESeq analysis

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

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
```

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
```

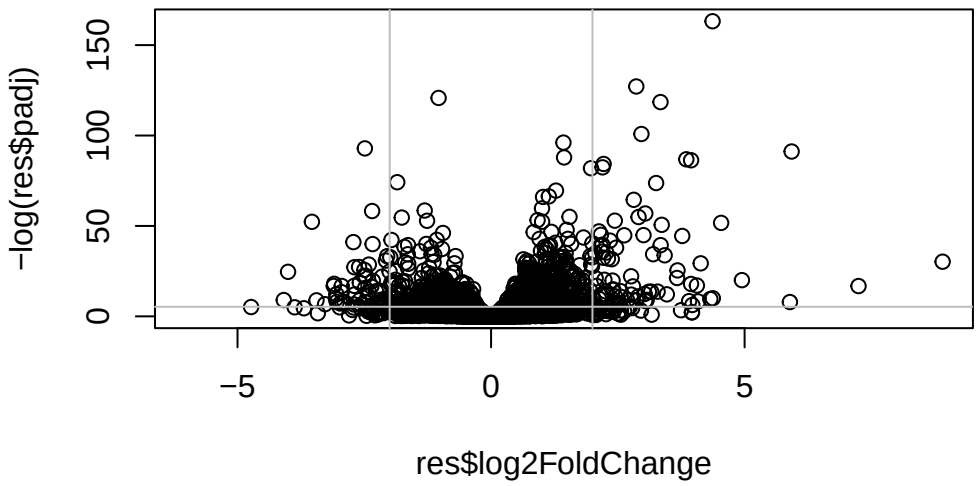
I will get the results from dds with the `results()` function:

```
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
```

Make a common overall results figure from this analysis. This designed to keep our inner biologist and inner stats nerd happy-it plots fold-change vs P-value

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



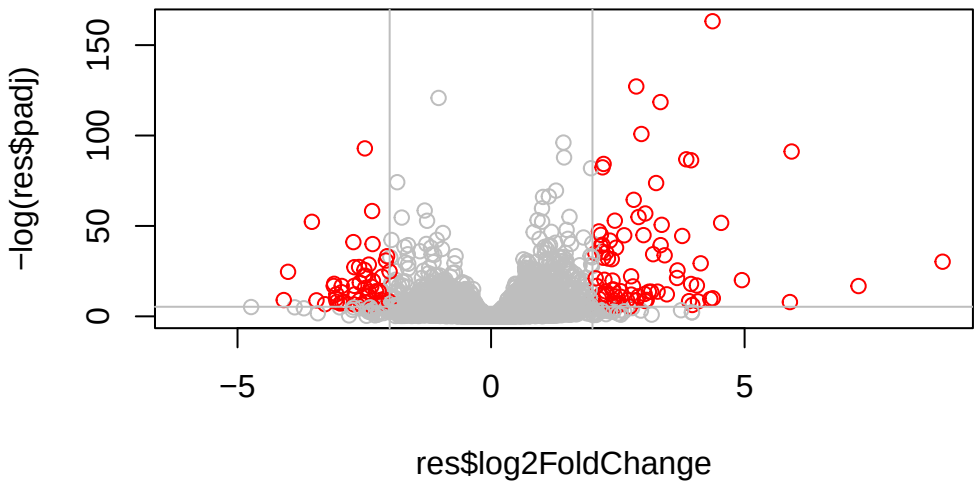

```
log(0.0000005)
```

```
[1] -14.50866
```

Add some color to this plot:

```
mycols <- rep("gray",nrow(res))
mycols[res$log2FoldChange > 2] <- "red"
mycols[res$log2FoldChange < -2] <- "red"
mycols[res$padj>0.005] <- "gray"


plot(res$log2FoldChange,-log(res$padj),col=mycols)
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 this up next day and add **annotation** (i.e. what are these genes of interest) and do **pathway analysis** (i.e. what biology) are they known to be involved with.

##Annotation I need to translate our gene identifiers "ENSG0000..." into gene names that the rest of the world can understand.

To do this annotion I will use the **AnnotationDbi** package. I can install this with `BiocManager::install()`

```
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)
```

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
```

And "ENTREZID"

```
res$entrez <- 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      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
```

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

Lets 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)
library(gage)
library(gageData)

data(kegg.sets.hs)


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. It just want a vector of importance. For RNASeq data like we have this is our log2FC values.

```
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
```

Now, let's run the gage pathway analysis.

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

What is in this `keggres` object?

```
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
```

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

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

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

Info: Working in directory /Users/sophiawang/Downloads/BIMM 143/Class 13: Transcriptomics an

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

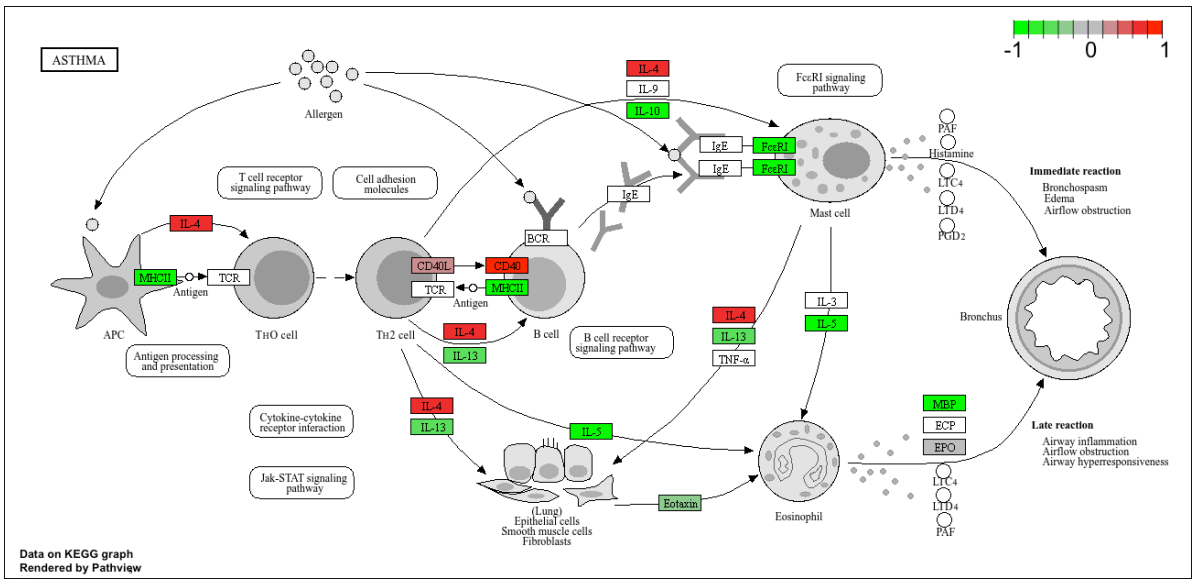


Figure 1: Asthma pathway with my DEGs