Class 12: RNASeq analysis

Fan Wu(PID: A15127541)

Table of contents

Background

Today we will analyze some RNASeq data from Himes et al. on the effects of a common steriod(dexamethasone) on airway smooth muscle cells (ASM cells).

Our starting point is the "counts" data and "metadata" that contain the count values for each gene in their different experiments (i.e. cell lines with or without the drug).

Data Import

```
# Complete the missing code
counts <- read.csv("airway_scaledcounts.csv", row.names=1)
metadata <-  read.csv("airway_metadata.csv")
```

Let's have a wee peak at these objects:

```
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
```

Q. How many different experiments (column in counts or rows in metadata) are there?

```
ncol(counts)
```

```
[1] 8
```

```
nrow(metadata)
```

```
[1] 8
```

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

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

```
control treated
      4       4
```

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

```
[1] 4
```

Toy differential gene expression

To start our analysis, let's calculate the mean counts for all genes in the "control" experiments.

1. Extract all "control" columns from the `counts` object
2. Calculate the mean for all rows (i.e. genes) of these "control" column 3-4 Do the same for "treated"
3. compare these `control.mean` and `treated.mean` values

```
#Step 1
control.inds <- metadata$dex == "control"
# control.inds <- grep("control", metadata$dex)
control.counts <- counts[, control.inds]

dim(control.counts)
```

```
[1] 38694     4
```

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

`rowMeans()`

```
# calculate mean of each row, i.e. gene, in "control"
control.means <- rowMeans(control.counts)
```

> 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)

```
# Step 3-4
treated.inds <- metadata$dex == "treated"

treated.counts <- counts[, treated.inds]

treated.means <- rowMeans(treated.counts)
```

Store these together for each of bookkeeping as `meancounts`

```
meanscounts <- data.frame(control.means, treated.means)
head(meanscounts)
```

```
                control.means treated.means
ENSG00000000003        900.75        658.00
ENSG00000000005          0.00          0.00
ENSG00000000419        520.50        546.00
ENSG00000000457        339.75        316.50
ENSG00000000460         97.25         78.75
ENSG00000000938          0.75          0.00
```

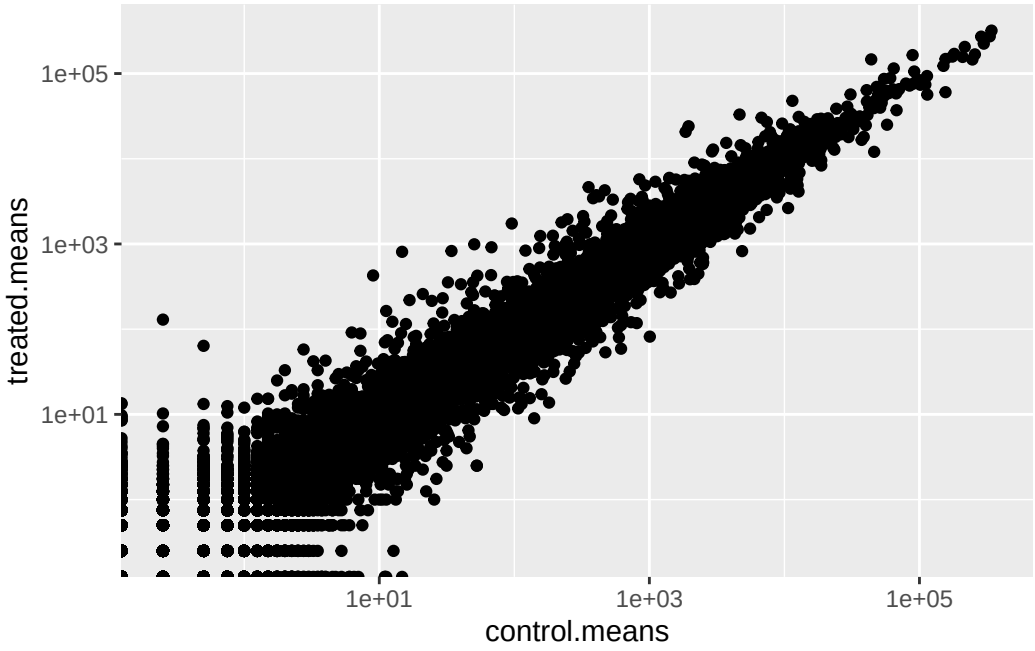
> Q5 (a). Create a scatter plot showing the mean of the treated samples against the mean of the control samples. Your plot should look something like the following.

Make a plot of control vs treated mean values for all genes plot per point per gene

```
library(ggplot2)
ggplot(meanscounts, aes(x= control.means, y = treated.means)) +
  geom_point()+
  scale_x_log10() +
  scale_y_log10()
```

```
Warning in scale_x_log10(): log-10 transformation introduced infinite values.
```

```
Warning in scale_y_log10(): log-10 transformation introduced infinite values.
```

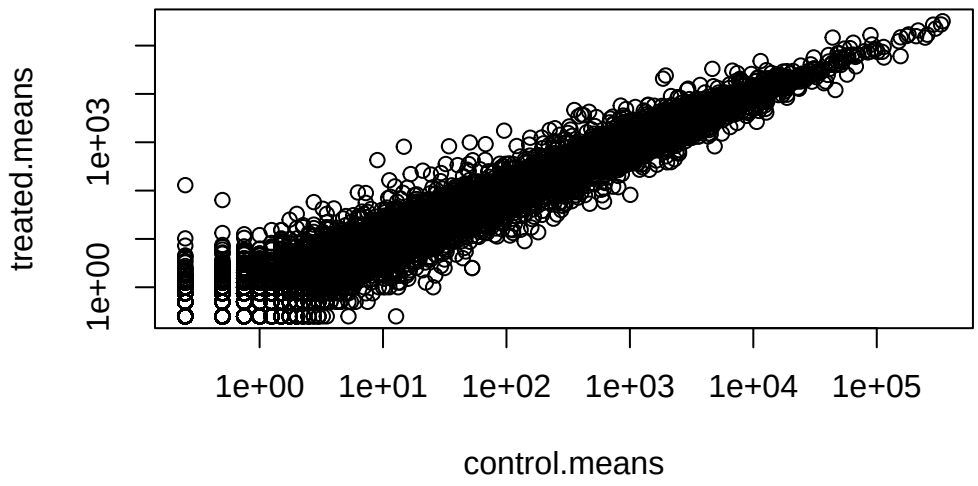


Make this a log plot

```
plot(meanscounts, 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 often talk about metrics like "log2 fold-change"

```
# control / treated
log2(10 / 10)
```

```
[1] 0
```

```
# log2(1) = 0, meaning no change, so `log2()` is useful in showing change
```

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

```
[1] -1
```

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

```
[1] 1
```

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

```
[1] 2
```

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

```
[1] -2
```

Let's calculate the **log2 fold-change** for our treated over control

```
meanscounts$log2fc <-
log2(meanscounts$treated.means /
  meanscounts$control.means)
```

```
head(meanscounts)
```

```
                control.means treated.means      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
```

A common "rule of thumb" is a log2 fold-change cutoff of +2 and -2 call genes "Up regulated" or "Down regulated", when log2(x/y) = +-2 means 4 times difference

Number of upregulated genes

```
sum(meanscounts$log2fc >= 2, na.rm= T)
```

```
[1] 1910
```

Number of downregulated genes at -2 threshold

```
sum(meanscounts$log2fc <= -2, na.rm= T)
```

```
[1] 2330
```

The above data is missing the **statistical significance** evaluation of these statistics: are these changes by chance or are they caused by the drug

DESeq2 analysis

Let's do this analysis properly and keep our inner stats nerd happy - i.e. are the differences we see btw/ drug and no drug significant, given the replicate experiments.

```
library(DESeq2)
```

For DESeq analysis, we need 3 things

1. count values (`contData`)
2. metadata telling us about the columns in `countData` (`colData`)
3. design of the experiment (i.e. what do you want to compare)

Our first function from DESeq2 will setup the input required for analysis by storing all these 3 things together.

```
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
```

```
# design = ~dex means this is what we are comparing
```

The main function in DESeq2 that runs the analysis is called `DESeq()`

```
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)
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
```

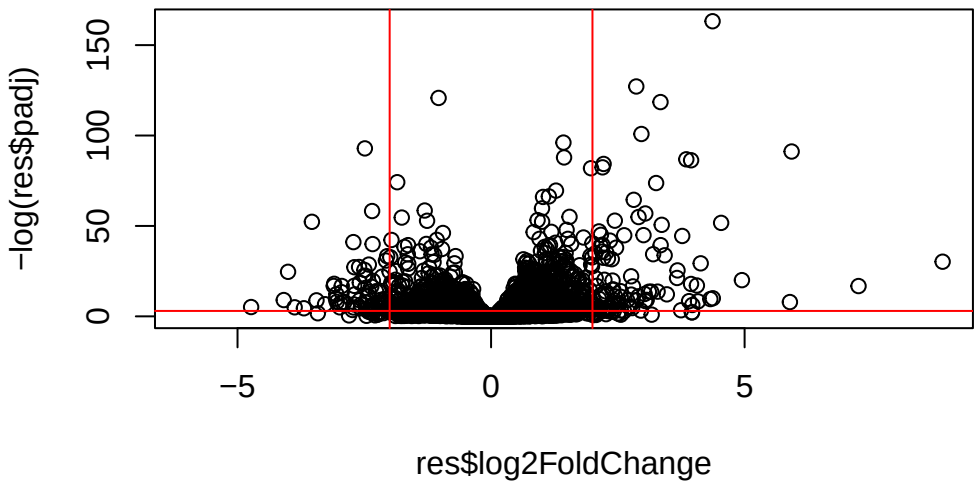
```
36000 * 0.05
```

```
[1] 1800
```

Volcano Plot

This is a common summary result figure from these types of experiments and plot the **log2 fold-change vs the adjusted p-value.**

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




```
log(0.1)
```

```
[1] -2.302585
```

```
log(0.001)
```

```
[1] -6.907755
```

Save our results

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

Add gene annotation

To help make sense of our results, and communicate them to other folks we need to add some more annotation to our main `res` object

We will use two bioconductor packages to first map IDs to different formats, including the classic gene "symbol" gene name.

I will install these with the following commands: `BiocManager::install("AnnotationDbi")` `BiocManager::install("org.Hs.eg.db")`

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

Let's see what is in `org.Hs.eg.db` w/ the `columns()` function:

```
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"
```

We can translate or "map" IDs between any of these 26 databases using the `mapIds()` function

```
res$symbol <- mapIds(keys = row.names(res),# current IDs
                     keytype = "ENSEMBL", # the format of our IDs
                     x = org.Hs.eg.db ,# where to get mappings from
                     column = "SYMBOL" # the format/DB to map to
       )
```

```
'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 the mappings for "GENENAME" and "ENTREZID" (NCBI identifier), a and store as `res$genename` and `res$entrez`

```
res$genename <- mapIds(keys = row.names(res),# current IDs
                    keytype = "ENSEMBL", # the format of our IDs
                    x = org.Hs.eg.db ,# where to get mappings from
                    column = "GENENAME" # the format/DB to map to
       )
```

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

```
res$entrez <- mapIds(keys = row.names(res),# current IDs
                    keytype = "ENSEMBL", # the format of our IDs
                    x = org.Hs.eg.db ,# where to get mappings from
                    column = "ENTREZID" # the format/DB to map to
       )
```

```
'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
```

Pathway analysis

Depend on the biological question we are interested in, we will use different functional set databases

Take our different expressed genes, check for overlap of these genes w/ different pathway

There are lots of bioconductor packages to do this type of analysis, now let's just try one called **GAGE** again we need to install this if we don't have it already

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

To use **GAGE** I need two things

- a named vector of fold-change values for our DEGs (our geneset of interest)
- a set of pathways or genesets to use for annotation.

```
x = c("david" = 5, "barry" = 10)
x
```

```
david barry
    5    10
```

```
names(x)
```

```
[1] "david" "barry"
```

```
names(x) <-c("low", "high")
x
```

```
 low high
   5   10
```

```
foldchanges <- res$log2FoldChange
#name by entrezid, as KEGG only speak in entrezid, not genename
names(foldchanges) <- res$entrez
head(foldchanges)
```

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

```
#run gageData for pathway
data("kegg.sets.hs")

# Get the results
keggres = gage(foldchanges, gsets=kegg.sets.hs)
```

In our results object we have:

```
attributes(keggres)
```

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

```
head(keggres$less, 5)
```

```
                                                    p.geomean stat.mean
hsa05332 Graft-versus-host disease                 0.0004250461 -3.473346
hsa04940 Type I diabetes mellitus                  0.0017820293 -3.002352
hsa05310 Asthma                                    0.0020045888 -3.009050
hsa04672 Intestinal immune network for IgA production 0.0060434515 -2.560547
hsa05330 Allograft rejection                       0.0073678825 -2.501419
                                                          p.val      q.val
hsa05332 Graft-versus-host disease                 0.0004250461 0.09053483
hsa04940 Type I diabetes mellitus                  0.0017820293 0.14232581
hsa05310 Asthma                                    0.0020045888 0.14232581
```

```
hsa04672 Intestinal immune network for IgA production 0.0060434515 0.31387180
hsa05330 Allograft rejection                          0.0073678825 0.31387180
                                                      set.size         exp1
hsa05332 Graft-versus-host disease                          40 0.0004250461
hsa04940 Type I diabetes mellitus                           42 0.0017820293
hsa05310 Asthma                                             29 0.0020045888
hsa04672 Intestinal immune network for IgA production       47 0.0060434515
hsa05330 Allograft rejection                                36 0.0073678825
```

Let's look at one of these pathways w/ our genes colored up so we can see the overlap

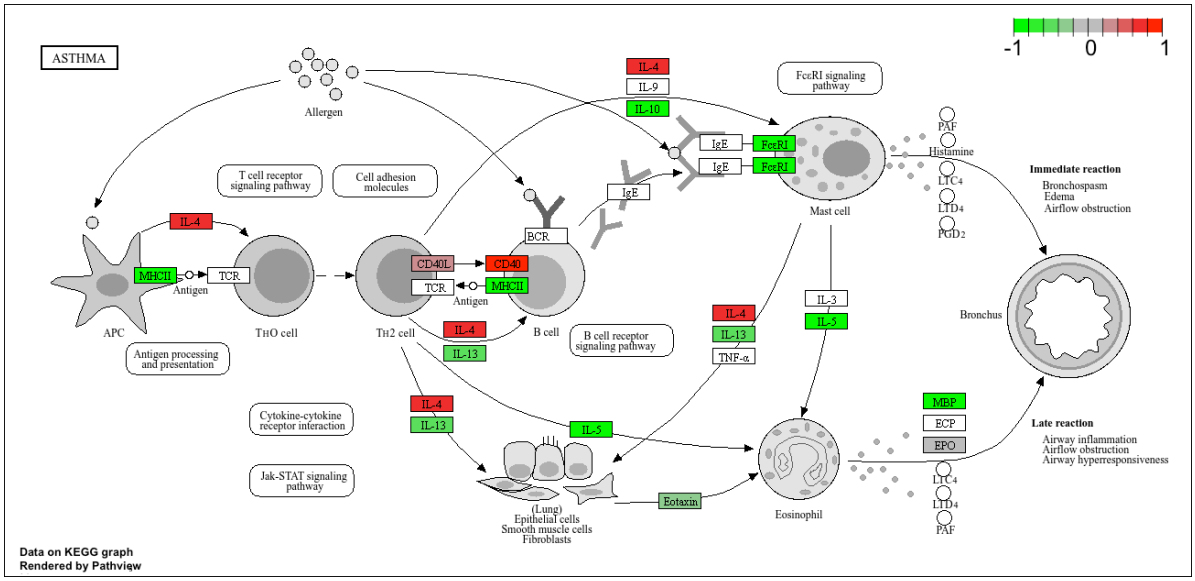
```
pathview(pathway.id = "hsa05310" ,gene.data = foldchanges)
```

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

```
Info: Working in directory /Users/fanwu/Desktop/BIMM143/FA25_Class12
```

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

Add this pathway figure to our lab report



Save our main results

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