Class 12: RNASeq analysis

Michael Romero (A18135877)

Table of contents

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

Are 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")
```

Lets 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(metadata)
```

```
[1] 8
```

```
ncol(counts)
```

```
[1] 8
```

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

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

```
[1] 4
```

Toy differential gene expression

To star tour analysis let's calculat ehte 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" columns

3-4. Do the same for "treated" 5. compare these `control.mean` and `treated.mean` values.

```
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
```

```
control.inds <- metadata$dex=="control"
control.inds
```

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

```
control.counts <- counts[,control.inds]
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
```

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

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

A data frame to consolidate the data.

```
treated.inds <- metadata$dex=="treated"
treated.inds
```

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

```
treated.counts <- counts[, treated.inds]
head(treated.counts)
```

```
                SRR1039509 SRR1039513 SRR1039517 SRR1039521
ENSG00000000003        486        445       1097        604
ENSG00000000005          0          0          0          0
ENSG00000000419        523        371        781        509
ENSG00000000457        258        237        447        324
ENSG00000000460         81         66         94         74
ENSG00000000938          0          0          0          0
```

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)

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

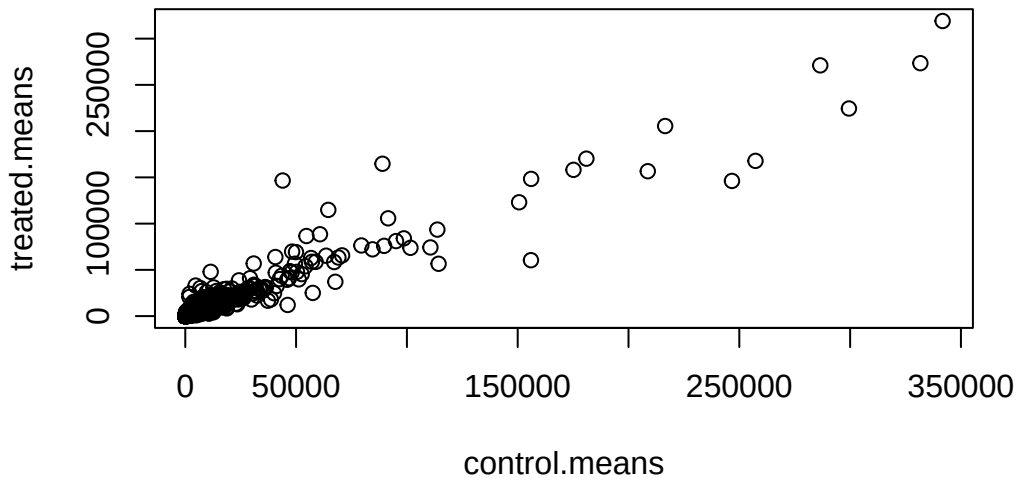
Compare the `control.means` and `treated.means`

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

```
                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.

```
plot(meancounts)
```



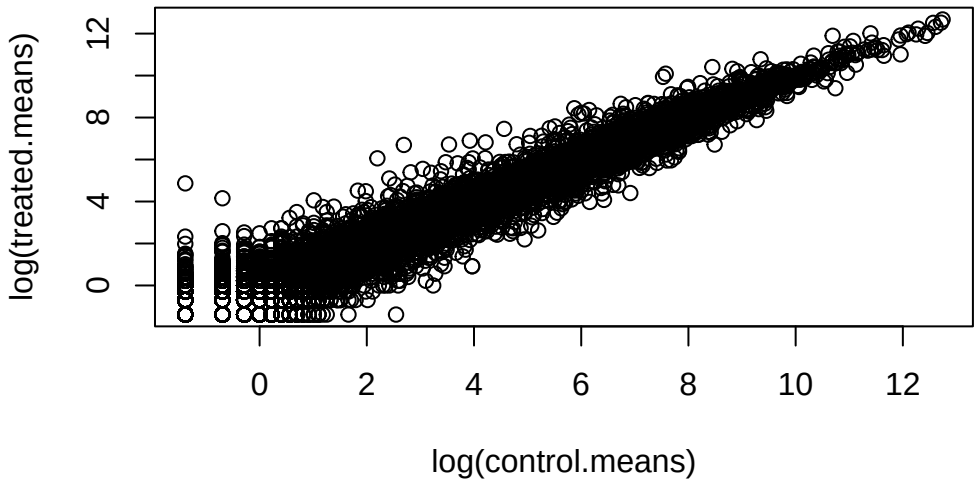
Q5 (b).You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot?

geom_point()

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

plot(log(x), log(y))

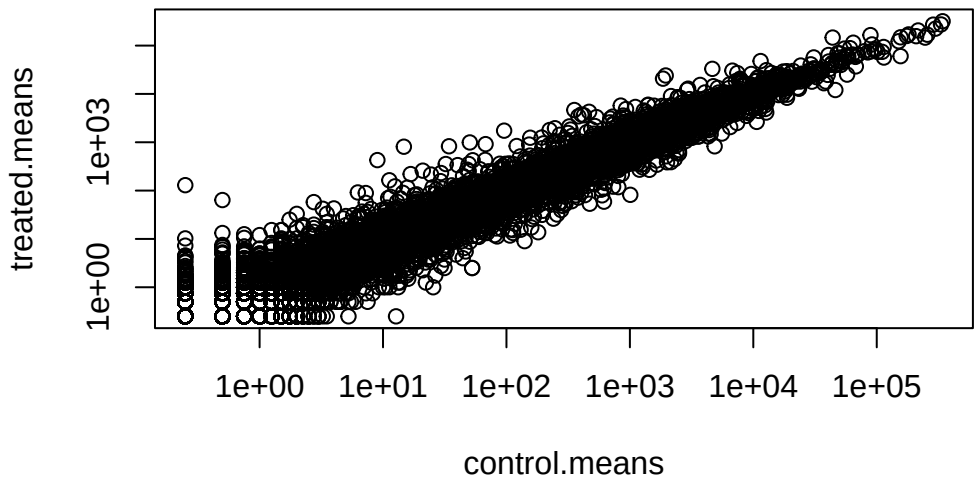
```
plot(log(control.means), log(treated.means))
```




```
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 often talk metrics like "log2 fold-change"

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

```
[1] 0
```

```
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 mean counts.

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

```
meancounts2 <- data.frame(control.means, treated.means, log2fc)
head(meancounts2)
```

```
                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
```

> 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 returns row and column values that are TRUE, and making the first column of the output unique allows us to use only the rows and remove duplicates from the data.

A common "rule of thumb" is a log2 fold change cutoff of +2 and -2 to call genes "Up regulated" or "Down regulated".

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

Number of "up" genes

```
sum(meancounts2$log2fc > +2, na.rm=T)
```

```
[1] 1846
```

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

Number of "down" genes at -2 threshold

```
sum(meancounts2$log2fc <= (-2), na.rm=T)
```

```
[1] 2330
```

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

Yes I trust these results since there is a similar amount of extreme values on both ends of the data set.

##DESeq2 analysis

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

```
library(DESeq2)
```

For DESeq analysis we need three things

- count values (`contData`)
- metadata telling us about the columns in `countData` (`colData`)
- 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
```

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
```

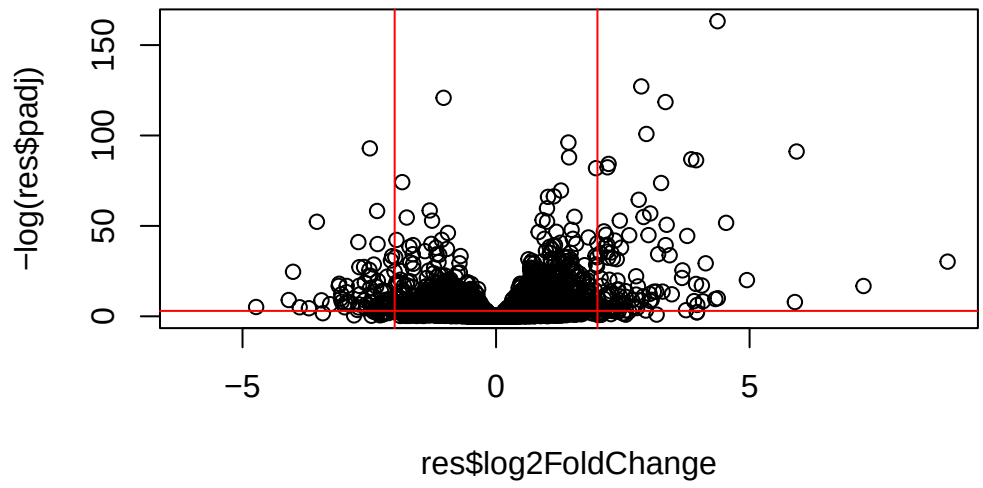
```
res <- results(dds)
head(res)
```

```
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      -0.350703  0.168242 -2.084514 0.0371134
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160       0.206107  0.101042  2.039828 0.0413675
ENSG00000000457 322.664844       0.024527  0.145134  0.168996 0.8658000
ENSG00000000460  87.682625      -0.147143  0.256995 -0.572550 0.5669497
ENSG00000000938   0.319167      -1.732289  3.493601 -0.495846 0.6200029
                     padj
                <numeric>
ENSG00000000003  0.163017
ENSG00000000005        NA
ENSG00000000419  0.175937
ENSG00000000457  0.961682
ENSG00000000460  0.815805
ENSG00000000938        NA
```

Volcano Plot

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

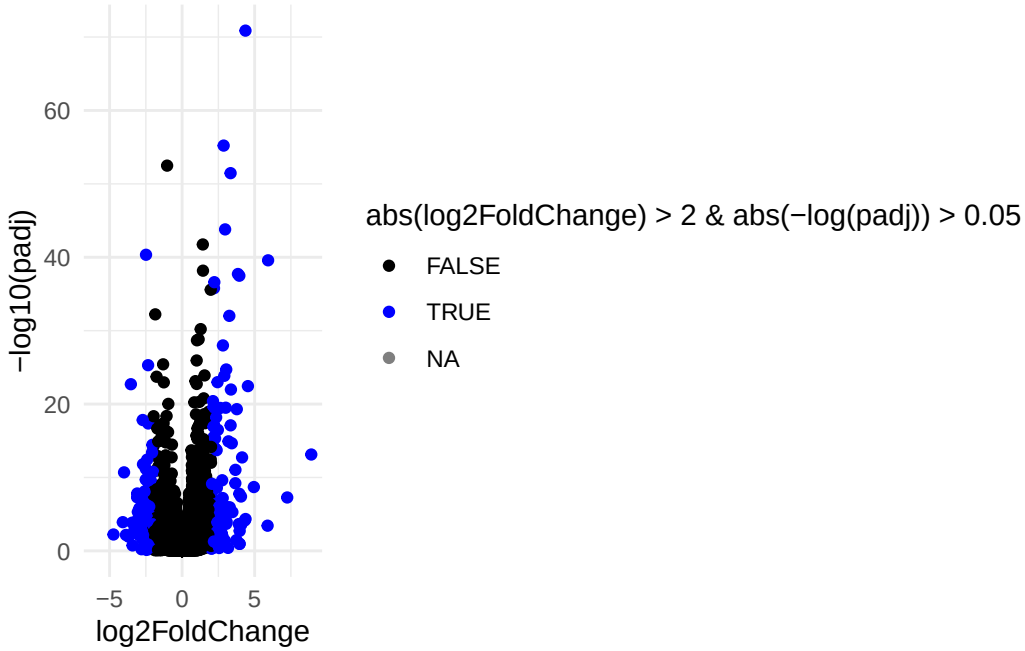
```
plot(res$log2FoldChange, -log(res$padj))
abline(v=c(-2,2), col="red")
abline(h=-log(0.05), col="red")
```




```
library(ggplot2)
```

```
ggplot(res, aes(x = log2FoldChange, y = -log10(padj),
                color = abs(log2FoldChange) > 2 & abs(-log(padj)) > 0.05)) +
  geom_point() +
  scale_color_manual(values = c("black", "blue")) +
  theme_minimal()
```

```
Warning: Removed 23549 rows containing missing values or values outside the scale range
(`geom_point()`).
```




```
log(0.1)
```

```
[1] -2.302585
```

```
log(0.000001)
```

```
[1] -13.81551
```

#Save our Results

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