class12

Joshua Martin (PID: A18545389)

Table of contents

Background

Today we will analyze some RNASeq data from Himes et al. on the effects of a common steroid (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 experiment (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
```

Q. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

```
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
```

```
ncol(counts)
```

```
[1] 8
```

```
nrow(counts)
```

```
[1] 38694
```

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

```
metadata$dex == "control"
```

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

```
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 for the `counts` object
2. Calculate the mean for lal rows (i.e. genes) of these "control" columns

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

```
control.inds <- metadata$dex == "control"
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 <- rowSums( control.counts )/4
head(control.means)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         900.75            0.00          520.50          339.75           97.25
ENSG00000000938
           0.75
```

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

use rowMeans instead of dividing by 4 after setting by group size.

control.mean <- rowMeans(counts[, metadata$dex == "control", drop = FALSE])treated.mean <- rowMeans(counts[, metadata$dex == "treated", drop = FALSE])sd

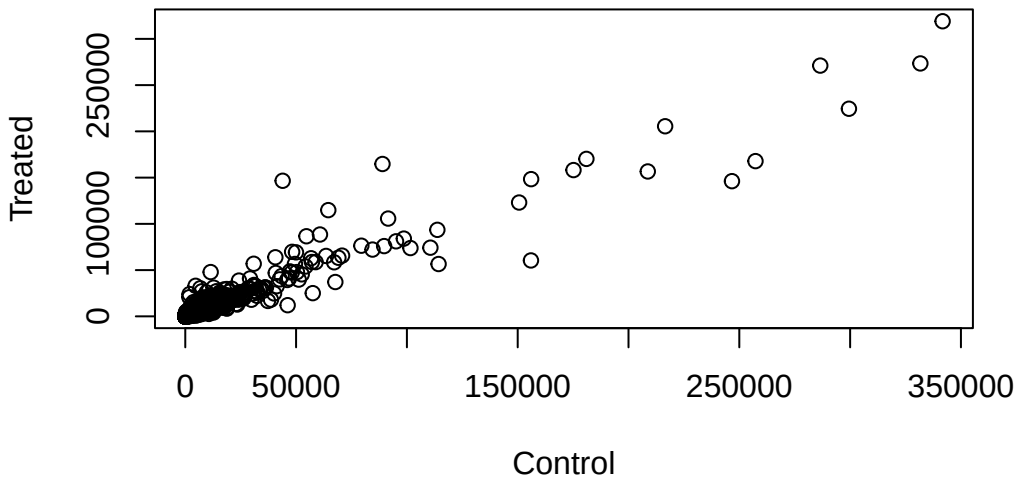
```
treated.inds <- metadata$dex == "treated"

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

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

Store these together for ease of bookkeeping as `meancounts`

```
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
```

```
plot(meancounts[,1],meancounts[,2], xlab="Control", ylab="Treated")
```

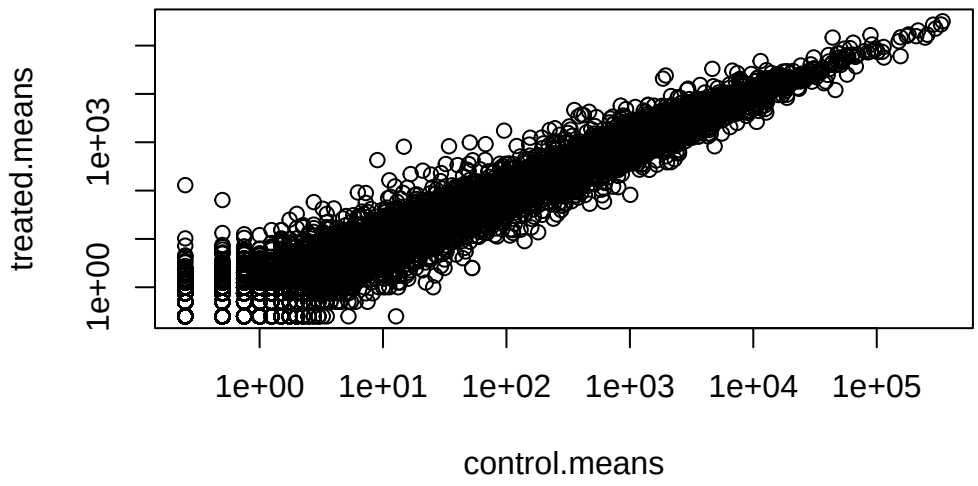


Make this a log log plot

```
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 liike "log2 fold-change"

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

```
[1] 0
```

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

```
[1] -1
```

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

```
[1] 1
```

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

```
[1] -2
```

```
zero.vals <- which(meancounts[,1:2] == 0, arr.ind = TRUE)
to.rm <- unique(zero.vals[,1])
mycounts <- meancounts[-to.rm, ]
```

> 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?

returns both the row and column positions of zeros. we take the row first column to remove genes with any zero counts, and unique() prevents removing the same row multiple times.

Let's calculate the log2 fold change for our treated over control mean counts.

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

```
head(meancounts)
```

```
                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 of +2 and -2 to call genes "Up regulated" or "Down regulated".

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

```
[1] 1846
```

Number of "down" genes at -2 threshold

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

```
[1] 2330
```

The above data is missing a statistical significance test.

up.ind <- mycounts$log2fc > 2 down.ind <- mycounts$log2fc < (-2) ## DESeq2 analysis

```
up.ind <- mycounts$log2fc > 2
down.ind <- mycounts$log2fc < (-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?

```
sum(mycounts$log2fc > 2, na.rm = TRUE)
```

```
[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?

```
sum(mycounts$log2fc < -2, na.rm = TRUE)
```

```
[1] 0
```

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

No. they are calculated from unnormalized mean counts and not accounting for variance, may include false positives/negativs.

Let's do this analysis properly and keep our inner stats nerd happy = i.e. are the differences we see between drug and no drug statistically 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`)

Our first function from DESeq 2 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
```

```
results(dds)
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 38694 rows and 6 columns
                 baseMean log2FoldChange     lfcSE      stat    pvalue
                <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003  747.1942      -0.350703  0.168242 -2.084514 0.0371134
ENSG00000000005    0.0000             NA        NA        NA        NA
ENSG00000000419  520.1342       0.206107  0.101042  2.039828 0.0413675
ENSG00000000457  322.6648       0.024527  0.145134  0.168996 0.8658000
ENSG00000000460   87.6826      -0.147143  0.256995 -0.572550 0.5669497
...                    ...            ...       ...       ...       ...
ENSG00000283115  0.000000             NA        NA        NA        NA
ENSG00000283116  0.000000             NA        NA        NA        NA
ENSG00000283119  0.000000             NA        NA        NA        NA
ENSG00000283120  0.974916       -0.66825   1.69441 -0.394385  0.693297
ENSG00000283123  0.000000             NA        NA        NA        NA
```

```
                    padj
                <numeric>
ENSG00000000003  0.163017
ENSG00000000005        NA
ENSG00000000419  0.175937
ENSG00000000457  0.961682
ENSG00000000460  0.815805
...                   ...
ENSG00000283115        NA
ENSG00000283116        NA
ENSG00000283119        NA
ENSG00000283120        NA
ENSG00000283123        NA
```

```
36000 * 0.05
```

```
[1] 1800
```

Volcano Plot

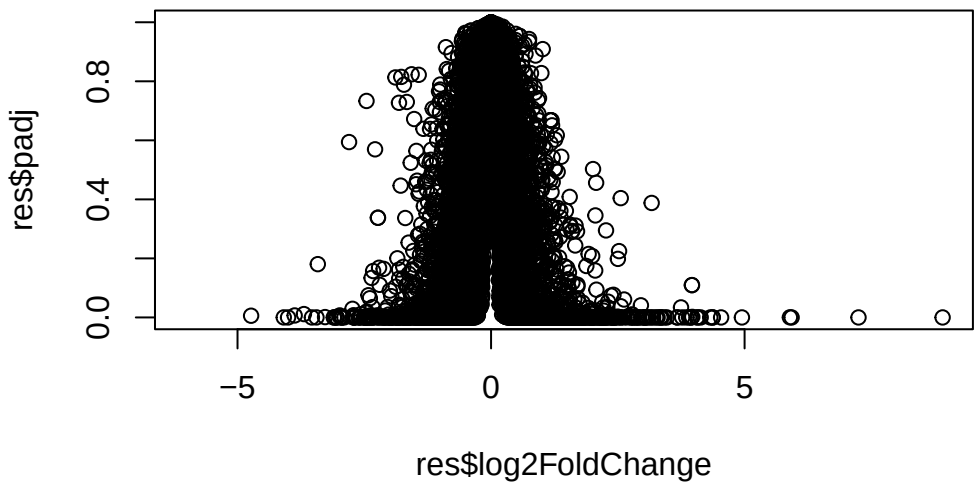
```
res <- results(dds)
head(res)
```

```
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
```

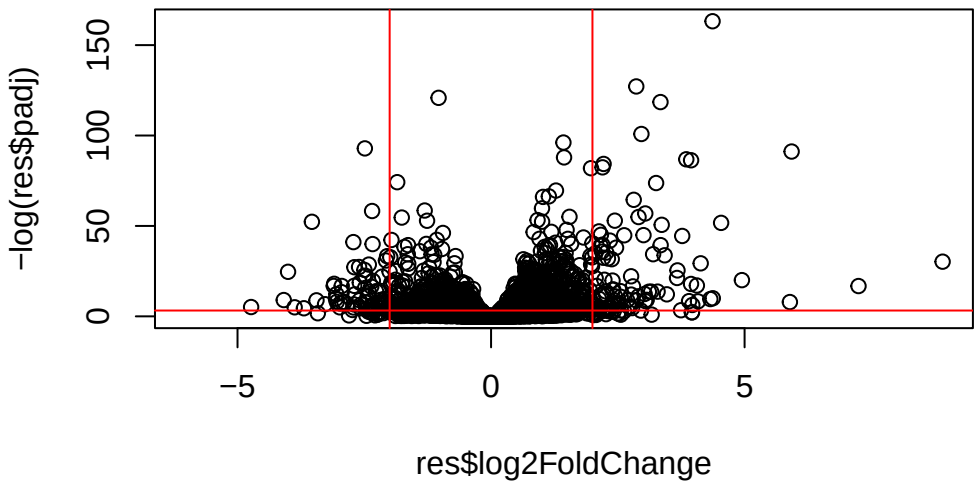
```
ENSG00000000460  0.815805
ENSG00000000938        NA
```

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$log2FoldChange, res$padj)
```




```
 plot( res$log2FoldChange,  -log(res$padj))
abline(v=c(-2,2), col="red")
abline(h=-log(0.04), col="red")
```



Save our results

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