Class 13: DESeq Analysis

Arshiya Zarmahd (PID: A16247996)

This week we are looking at differential expression analysis.

The data for this hands-on session comes from a published RNA-seq experiment where airway smooth muscle cells were treated with dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al., 2014).

Import/Read Data from Himes

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

Lets have a wee peak at this data!

```
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
```

Sanity Check on correspondence of counts and metadata

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

```
[1] TRUE
```

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

```
[1] FALSE
```

Q1. How many genes are in this dataset?

There are `nrow(counts)` genes in this dataset.

```
nrow(counts)
```

```
[1] 38694
```

Q2. How many control cell lines do we have?

There are `n.control` control cell lines in this dataset

```
n.control <- sum(metadata$dex == "control")
n.control
```

```
[1] 4
```

Extract and Summarize Control Data

To find out where the control samples are, we need the metadata.

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

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

Extract and Summarize the Treated (ie. Drug) Samples

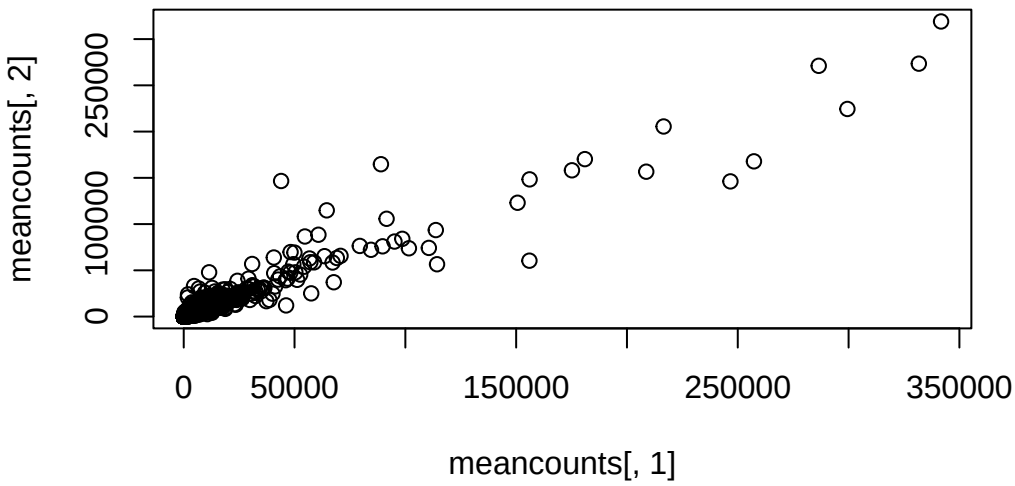
```
treated <- metadata[metadata$dex == "treated", ]
treated.counts <- counts[ , treated$id]
treated.mean <- rowMeans(treated.counts)
```

Store these results in a new data frame called `meancounts`.

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

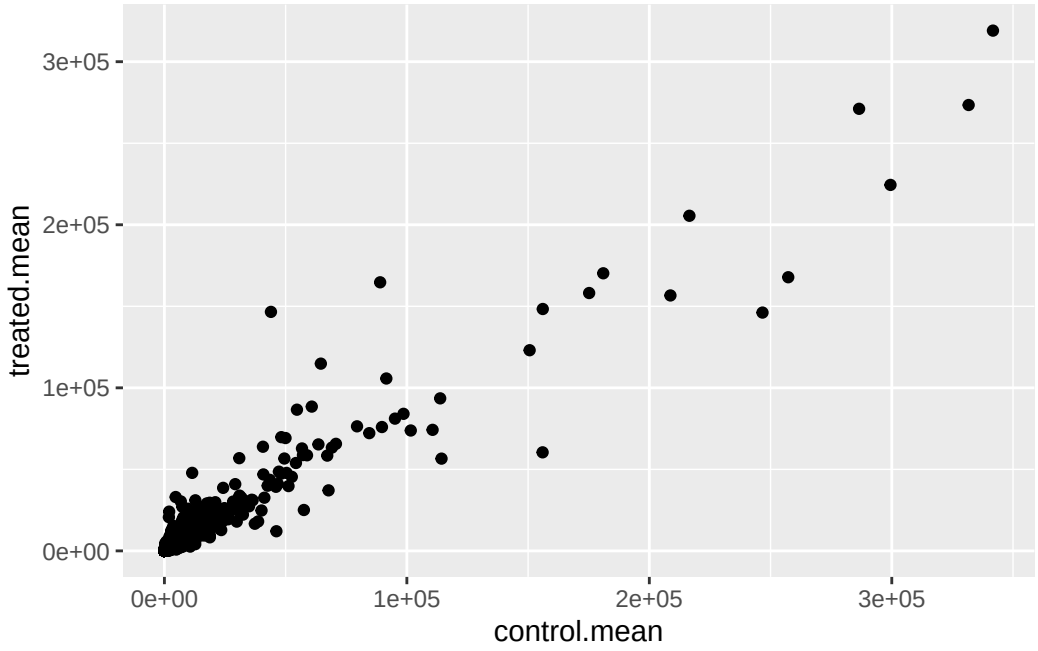
Lets make a plot to explore the results a little

```
plot(meancounts[,1], meancounts[,2])
```




```
library(ggplot2)
```

```
ggplot(meancounts) +
  aes(control.mean, treated.mean) +
  geom_point()
```

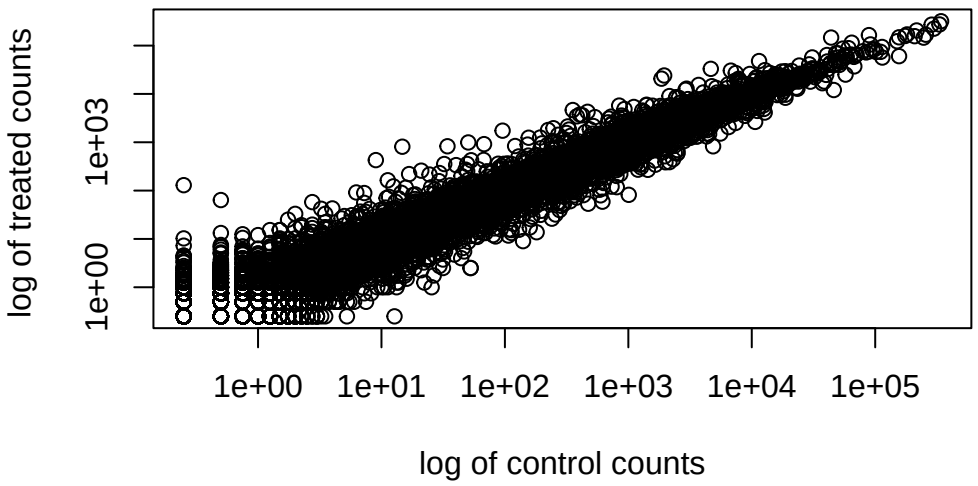


We will make a log-log plot to draw out this skewed data and see what is going on.

```
plot(meancounts[,1], meancounts[,2], log="xy",
     xlab="log of control counts",
     ylab="log of treated counts ")
```

```
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 use log2 transformations when dealing with this sort of data.

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

```
[1] 0
```

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

```
[1] 1
```

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

```
[1] -1
```

```
log2(80/20)
```

```
[1] 2
```

This log2 transformation has this nice property where if there is no change the log2 value will be zero and if its double the log2 value will be 1. If halved, it will be -1.

Lets add a log2 fold change column to our results.

```
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
```

We need to get rid of zero count genes that we can not say anything about.

```
head(meancounts[,1:2]==0)
```

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

```
head(which(meancounts[,1:2]==0, arr.ind=TRUE))
```

```
                row col
ENSG00000000005   2   1
ENSG00000004848  65   1
ENSG00000004948  70   1
ENSG00000005001  73   1
ENSG00000006059 121   1
ENSG00000006071 123   1
```

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

```
head(mycounts)
```

```
                control.mean treated.mean      log2fc
ENSG00000000003       900.75       658.00 -0.45303916
ENSG00000000419       520.50       546.00  0.06900279
ENSG00000000457       339.75       316.50 -0.10226805
ENSG00000000460        97.25        78.75 -0.30441833
ENSG00000000971      5219.00      6687.50  0.35769358
ENSG00000001036      2327.00      1785.75 -0.38194109
```

How many genes remain?

```
nrow(mycounts)
```

```
[1] 21817
```

Use Fold Change to See Up and Down Regulated Genes

A common threshold used for calling something differently expressed is a log2(FoldChange) of greater than 2 or less than -2. Lets filter the dataset both ways to see how many genes are up or down regulated.

Up-Regulated:

```
sum(mycounts$log2fc > 2)
```

```
[1] 250
```

Down-Regulated:

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

```
[1] 367
```

Do we trust these results? No, because we don't yet know if these changes are significant.

DESeq2 Analysis

```
library(DESeq2)
```

```
Loading required package: S4Vectors
```

```
Loading required package: stats4
```

```
Loading required package: BiocGenerics
```

```

Attaching package: 'BiocGenerics'
```

```
The following objects are masked from 'package:stats':

    IQR, mad, sd, var, xtabs
```

```
The following objects are masked from 'package:base':

    anyDuplicated, aperm, append, as.data.frame, basename, cbind,
    colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
    get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
    match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
    Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
    table, tapply, union, unique, unsplit, which.max, which.min
```

```

Attaching package: 'S4Vectors'
```

```
The following object is masked from 'package:utils':

    findMatches
```

```
The following objects are masked from 'package:base':

    expand.grid, I, unname
```

```
Loading required package: IRanges
```

```
Loading required package: GenomicRanges

Loading required package: GenomeInfoDb

Loading required package: SummarizedExperiment

Loading required package: MatrixGenerics

Loading required package: matrixStats


Attaching package: 'MatrixGenerics'

The following objects are masked from 'package:matrixStats':

    colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
    colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
    colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
    colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
    colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
    colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
    colWeightedMeans, colWeightedMedians, colWeightedSds,
    colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
    rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
    rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
    rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
    rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
    rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
    rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
    rowWeightedSds, rowWeightedVars

Loading required package: Biobase

Welcome to Bioconductor

    Vignettes contain introductory material; view with
    'browseVignettes()'. To cite Bioconductor, see
    'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
Attaching package: 'Biobase'

The following object is masked from 'package:MatrixGenerics':

    rowMedians

The following objects are masked from 'package:matrixStats':

    anyMissing, rowMedians
```

```
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
```

```
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)
res
```

```
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     -0.3507030  0.168246 -2.084470 0.0371175
ENSG00000000005    0.0000             NA        NA        NA        NA
ENSG00000000419  520.1342      0.2061078  0.101059  2.039475 0.0414026
ENSG00000000457  322.6648      0.0245269  0.145145  0.168982 0.8658106
ENSG00000000460   87.6826     -0.1471420  0.257007 -0.572521 0.5669691
...                    ...            ...       ...       ...       ...
ENSG00000283115  0.000000             NA        NA        NA        NA
ENSG00000283116  0.000000             NA        NA        NA        NA
ENSG00000283119  0.000000             NA        NA        NA        NA
ENSG00000283120  0.974916      -0.668258   1.69456 -0.394354  0.693319
ENSG00000283123  0.000000             NA        NA        NA        NA
                     padj
                <numeric>
ENSG00000000003  0.163035
ENSG00000000005        NA
ENSG00000000419  0.176032
ENSG00000000457  0.961694
ENSG00000000460  0.815849
...                   ...
ENSG00000283115        NA
ENSG00000283116        NA
ENSG00000283119        NA
ENSG00000283120        NA
ENSG00000283123        NA
```

We can get some basic summary tallies using the `summary()` function.

```
summary(res, alpha=0.05)
```

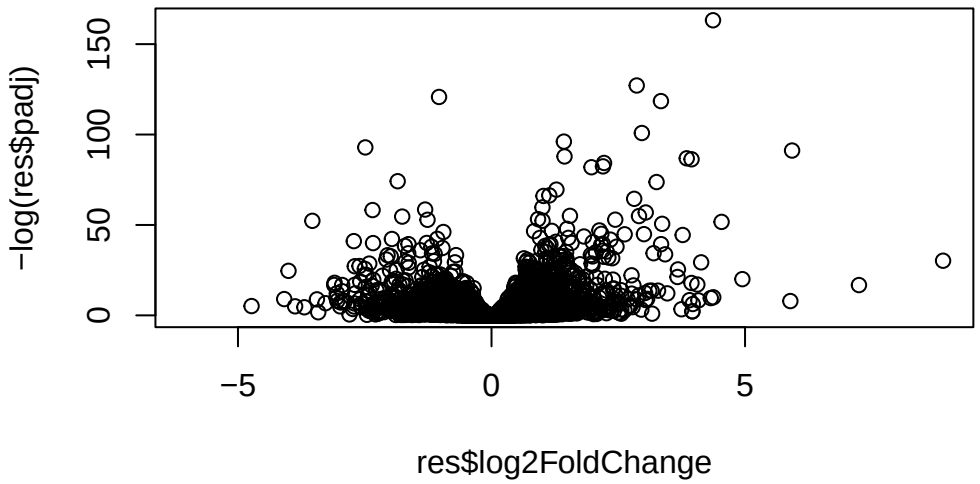
```
out of 25258 with nonzero total read count
adjusted p-value < 0.05
LFC > 0 (up)       : 1242, 4.9%
LFC < 0 (down)     : 939, 3.7%
outliers [1]       : 142, 0.56%
low counts [2]     : 9971, 39%
(mean count < 10)
```

```
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

Volcano Plot

Make a summary of our results.

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



Finish by saving results.

```
write.csv(res, file="DESeq2_results.csv")
```