Class 11: Transcriptomics and the analysis of RNA-Seq data

Taylor F. (A59010460)

2/23/2022

Installing DESeq2

```
#Do this in the console in the future
#install.packages("BiocManager")
#BiocManager::install()

#We will also need the DESeq2 package
#BiocManager::install("DESeq2")

#NOTE: Answer NO to prompts to install from source or update

#Run library(DESeq2) in the console
```

Today we will run differential expression analysis of some published data from Himes et al. where the authors used a published RNA-seq experiment where airway smooth muscle cells were treated with dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects.

Importing countData and colData

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

#Preview the counts dataset
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
```

```r
#Determine how many genes are in this counts dataset
nrow(counts)
```

```
## [1] 38694
```

```r
#Determine how many control cell lines used
ncol(counts)
```

```
## [1] 8
```

```r
#Look at the metadata dataset
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
```

Question 1:

There are 38694 rows, which translate to genes, in the counts dataset.

Question 2:

There are 8 rows, which translate to individual cell lines, in the counts dataset.

Based on the metadata, it looks like we have four drug-treated and four control cell lines. Our first question is does the drug do anything?

First, we want to check if the metadata matches the counts data order:

```r
#Grab the id column values of the metadata dataset
metadata$id
```

```
## [1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"
## [6] "SRR1039517" "SRR1039520" "SRR1039521"
```

```r
#Grab the column names of the counts dataset
colnames(counts)
```

```
## [1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"
## [6] "SRR1039517" "SRR1039520" "SRR1039521"
```

```
#Ask if these values are equivalent (all returned values should be TRUE)
metadata$id == colnames(counts)
```

```
## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
#Alternative method
all(metadata$id == colnames(counts))
```

```
## [1] TRUE
```

```
#Fancy method
if(all(metadata$id == colnames(counts))) {cat("Let's do this!")}
```

```
## Let's do this!
```

Next, we want to separate the two conditions (control and treated) from "counts" and use a summary statistic to make comparison easier.

```
#Control first
#We need to find the ID associated with control conditions (i.e. which id columns also have dex == "con
control.inds <- metadata[metadata$dex == "control","id"]
control.inds
```

```
## [1] "SRR1039508" "SRR1039512" "SRR1039516" "SRR1039520"
```

```
#We now use these indices to search through counts and finds these columns
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
```

```
#Question 4: We can now access the treated values, too
treated.inds <- metadata[metadata$dex == "treated","id"]
treated.inds
```

```
## [1] "SRR1039509" "SRR1039513" "SRR1039517" "SRR1039521"
```

```
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
```

Find the mean count value for each row (i.e. gene). We could use the 'apply()' function or more simply the 'rowMeans()' function.

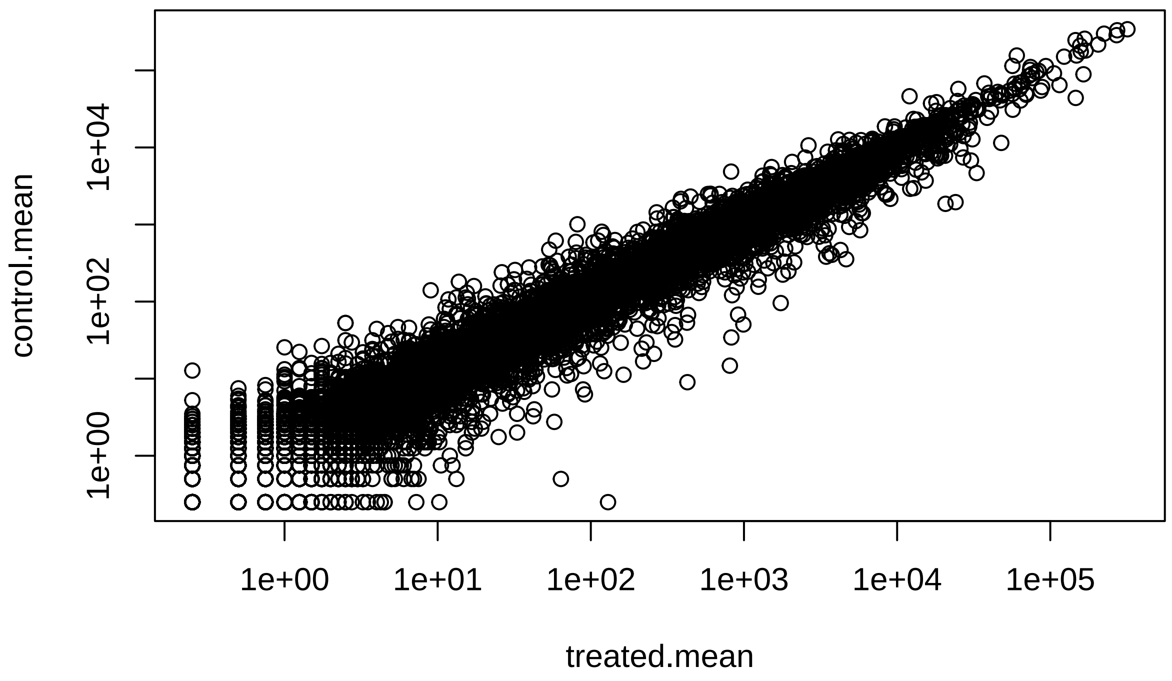
```
#Let's now find the means for each of these groups
control.mean <- rowMeans(control.counts)
treated.mean <- rowMeans(treated.counts)

#Question 5a: We can plot these mean values against one another
plot(treated.mean, control.mean, log = "xy")
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 15281 x values <= 0 omitted
## from logarithmic plot
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 15032 y values <= 0 omitted
## from logarithmic plot
```

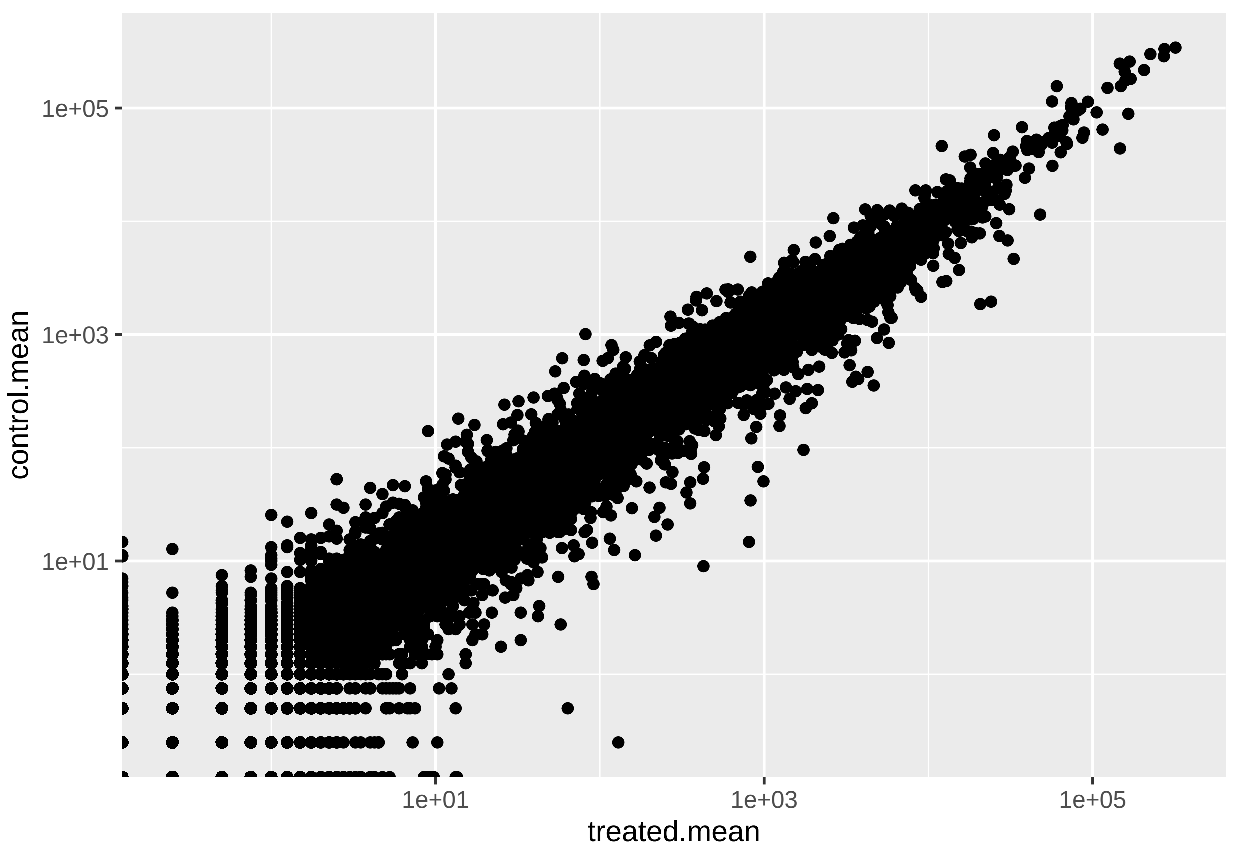
```
#Question 5b: We can also use ggplot
library(ggplot2)
```



```
ggplot(counts, aes(treated.mean, control.mean)) + geom_point() + scale_y_continuous(trans='log10') + sca
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```

```
## Warning: Transformation introduced infinite values in continuous x-axis
```



```
#Question 6: Try plotting both axes on a log scale. What is the argument to plot() that allows you to d
#log = "xy"
```

We often use log 2 transformation because it is easier to understand, visually.

```
#20/20
#log2(20/20)
#log2(40/20)
#log2(10/20)
#log2(80/20)

#Store the log2 fold change between treated and control groups
log2fc <- log2(treated.mean/control.mean)
```

Finding and filtering zero values

```
#We need to find and remove the genes that have zeros for values
log2fc[1:6]
```

```
## ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
##     -0.45303916             NaN      0.06900279     -0.10226805     -0.30441833
## ENSG00000000938
##            -Inf
```

```r
meancounts <- data.frame(control.mean, treated.mean, log2fc)
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
```

```r
z <- data.frame(x = c(1,2,0,4), y = c(1,2,0,0))

#Report which indices are TRUE and FALSE (i.e. sum to 0 or greater than 0) and gives index information
which(z == 0, arr.ind = TRUE)
```

```
##      row col
## [1,]   3   1
## [2,]   3   2
## [3,]   4   2
```

```r
unique(which(z == 0, arr.ind = TRUE)[,"row"])
```

```
## [1] 3 4
```

```r
#Apply this principle to the meancounts dataset
to.rm <- sort(unique(which(meancounts[,1:2] == 0, arr.ind = TRUE)[,"row"]))
mycounts <- meancounts[-to.rm,]
#mycounts
```

There are 21817 genes left over after removing zero values.

How many genes have a log2fc more than +2 (i.e. upregulated)?

```r
up.ind <- mycounts$log2fc > 2
down.ind <- mycounts$log2fc < (-2)
sum(up.ind)
```

```
## [1] 250
```

```r
sum(down.ind)
```

```
## [1] 367
```

There are 250 genes upregulated by at least 2log() and `rsum(down.ind)` genes downregulated by at least 2log().

But this approach does not treat fold-changes equally. There may be significant fold change increases or decreases in a gene expressed very low, and a comparatively large change that is not included in another gene that has a higher basal level of expression.

DESeq2

This approach will be the right way and will give us the stats.

```
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, 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 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
```

```
#Let's look at metadata again and set up the object that DESeq needs with the 'DESeqDataSetFromMatrix()
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
```

```
## class: DESeqDataSet
## dim: 38694 8
## metadata(1): version
## assays(1): counts
## rownames(38694): ENSG00000000003 ENSG00000000005 ... ENSG00000283120
##   ENSG00000283123
## rowData names(0):
## colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
## colData names(4): id dex celltype geo_id
```

```
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
```

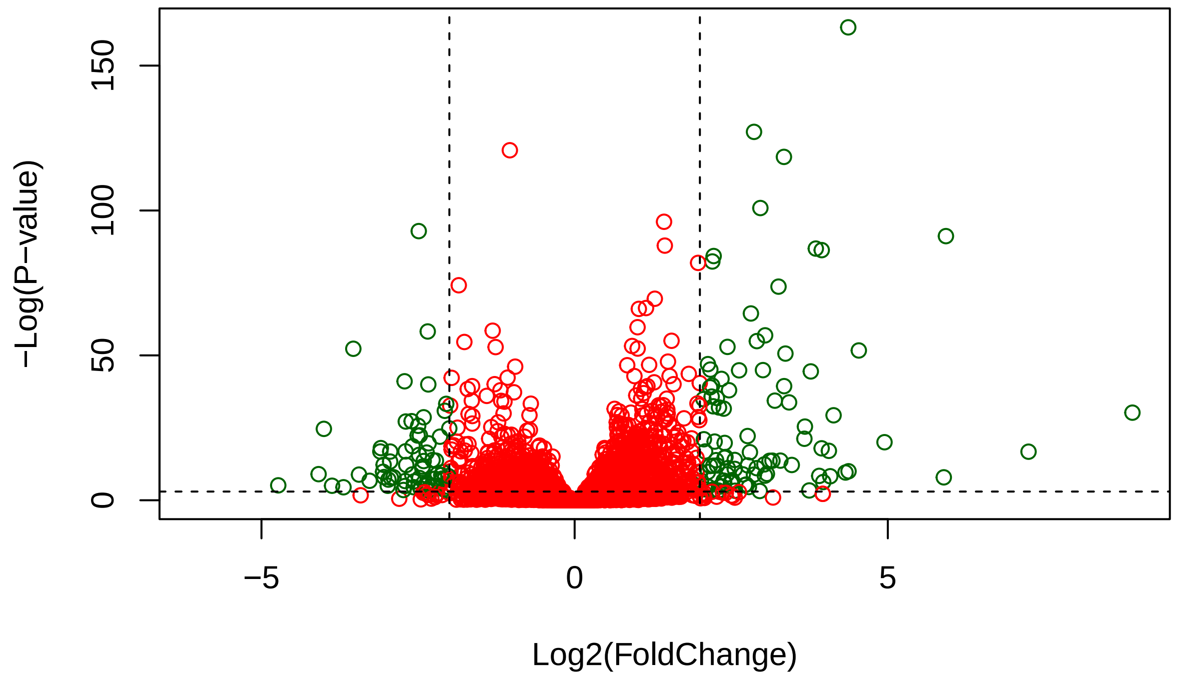
A main result figure

A common main result figure from this type of analysis is called a volcano plot. This is a plot of log2 fold change on the x axis vs. p-value.

```
#We can also color by significance and fold change
sigcols <- rep("gray", nrow(res))
sigcols[ abs(res$log2FoldChange) > 2 & (res$padj) <= 0.05 ]  <- "darkgreen"
sigcols[ abs(res$log2FoldChange) < 2  | (res$padj) > 0.05 ]  <- "red"

#Plot this data
plot(res$log2FoldChange, -log(res$padj), xlab = "Log2(FoldChange)",
      ylab = "-Log(P-value)", col = sigcols)

#Add significance lines to the plot
abline(v = c(-2,2), col = "black", lty = 2)
abline(h = -log(0.05), col = "black", lty = 2)
```


−Log(P−value)
Log2(FoldChange)