class11

Workspace Setup

```
#Get and load DESeq2 from Bioconductor for Data
#BiocManager::install("DESeq2")

library(BiocManager)
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
```

```
##
## Attaching package: 'IRanges'
```

```
## The following object is masked from 'package:grDevices':
##
##     windows

## 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
```

```
#Import countData and colData
counts <- read.csv("airway_scaledcounts.csv", row.names=1)
metadata <- read.csv("airway_metadata.csv")

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
```

```
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
```

```
#Double check that the columns of the countdata match the rows of the coldata; 'all()' funciton makes s
#'all()': all true -> TRUE, any false -> FALSE, all false -> FALSE
all(metadata$id == colnames(counts))
```

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

#Q1. There are 38694 genes in this dataset.

#Q2. There are 4 'control' cell lines.

Data pre-processing

```
#Separate data from the metadata table
metadata.ctrl <- metadata[metadata$dex == "control", ]
metadata.trtd <- metadata[metadata$dex == "treated", ]

#Extract control and treated counts
##DataSet[selection]$column_to_extract
control.ids <- metadata[metadata$dex == "control", ]$id
control.counts <- counts[,control.ids]
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
```

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

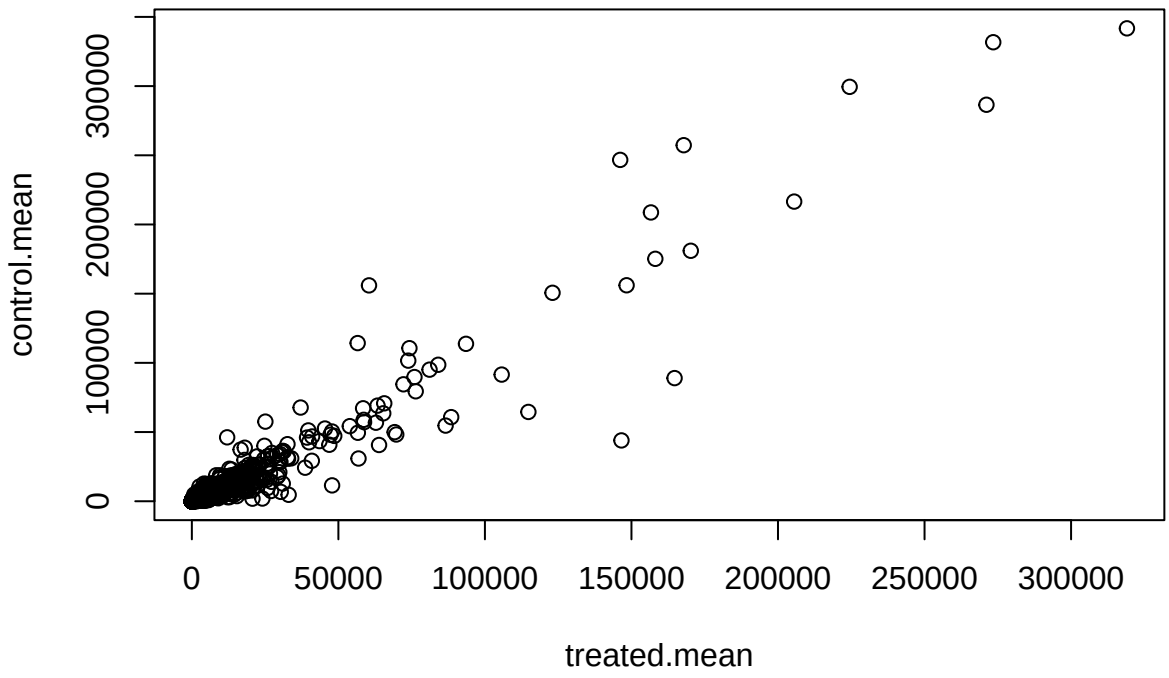
```
##                 SRR1039509 SRR1039513 SRR1039517 SRR1039521
## ENSG00000000003        486        445       1097        604
## ENSG00000000005          0          0          0          0
## ENSG00000000419        523        371        781        509
## ENSG00000000457        258        237    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

#Q3. You could make the code from the website more robust by using the mean() function instead of RowSums()/4, because in order to use the latter approach, you would need to find how many columns there are, whereas with the former, you will be able to get the answer without this research.

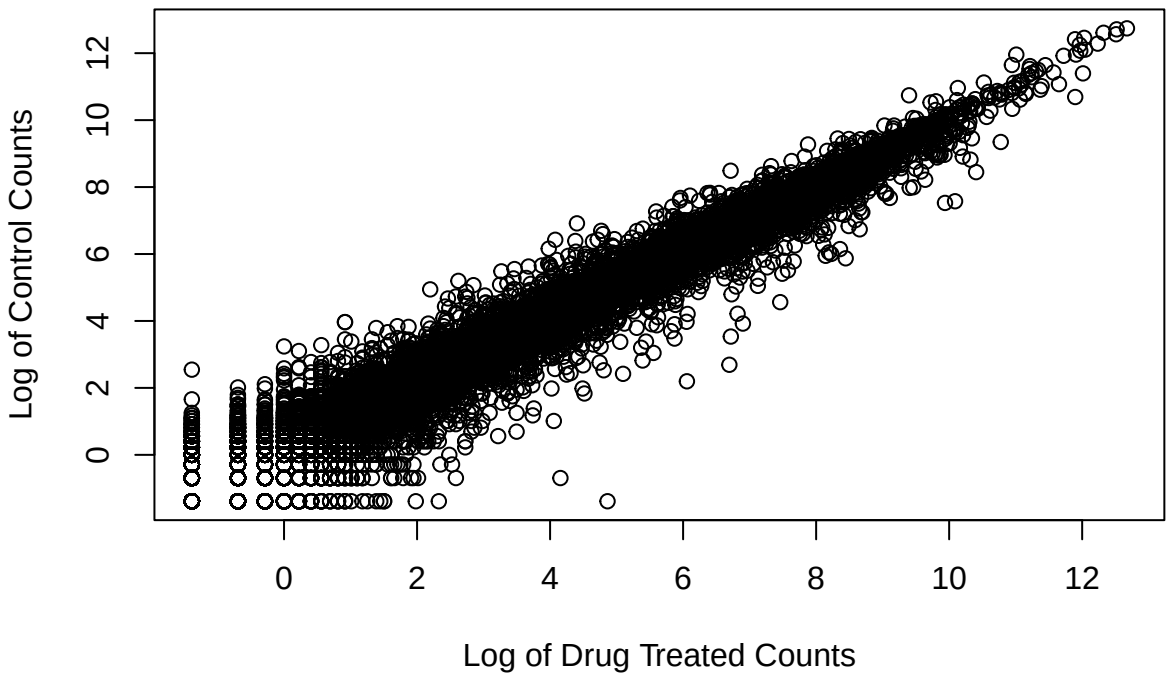
#Q4. Done.

Data that will be useful for us to look at

```
#Summarize & visualize extracted data
control.mean <- rowMeans(control.counts)
treated.mean <- rowMeans(treated.counts)

plot(treated.mean, control.mean) #Q5
```




```
plot(log(treated.mean), log(control.mean), #Q6
     xlab= "Log of Drug Treated Counts",
     ylab= "Log of Control Counts")
```




```
#ggplot function: scale_x_continuous(trans="log2")
```

```
#Calculate fold change between treated and untreated patients
#log2 of the fold change has better mathematical properties
treated.fold <- log2(treated.mean/control.mean)

#Organize Data
workingtable <- data.frame(control.mean, treated.mean, treated.fold)

#Remove Nonsensical Data
zero.vals <- which(workingtable[,1:2]==0, arr.ind=TRUE)

to.rm <- unique(zero.vals[,1])
workingtable2 <- workingtable[-to.rm,]
head(workingtable2)
```

```
##                 control.mean treated.mean treated.fold
## 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
```

#Q7. The purpose of the arr.ind is to make the output of the which() function into a table specifying the row and column of where the result is found, rather than a list that doesn't apply well onto a table. We

would then need to call the unique() function to ensure that we don't delete two separate rows for an instance where both the before and after values are 0.

```
#Find up-regulated & down-regulated genes
up.ind <- workingtable2$treated.fold > (2)
down.ind <- workingtable2$treated.fold < (-2)

#Determine number of up-regulated and down-regulated genes
sum(up.ind) #Q8
```

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

```
sum(down.ind) #Q9
```

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

#Q8. There are 250 up-regulated genes.

#Q9. There are 367 donw-regulated genes.

#Q10. These results are good, but they need to be determined whether or not they are actually statistically significant. The mean is also a single-number summary that can hide a lot of detail within itself.

DESeq2 Analysis

```
#Format data into input for DESeq2
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
```

```
#Run the program to get p-values
dds <- DESeq(dds)
```

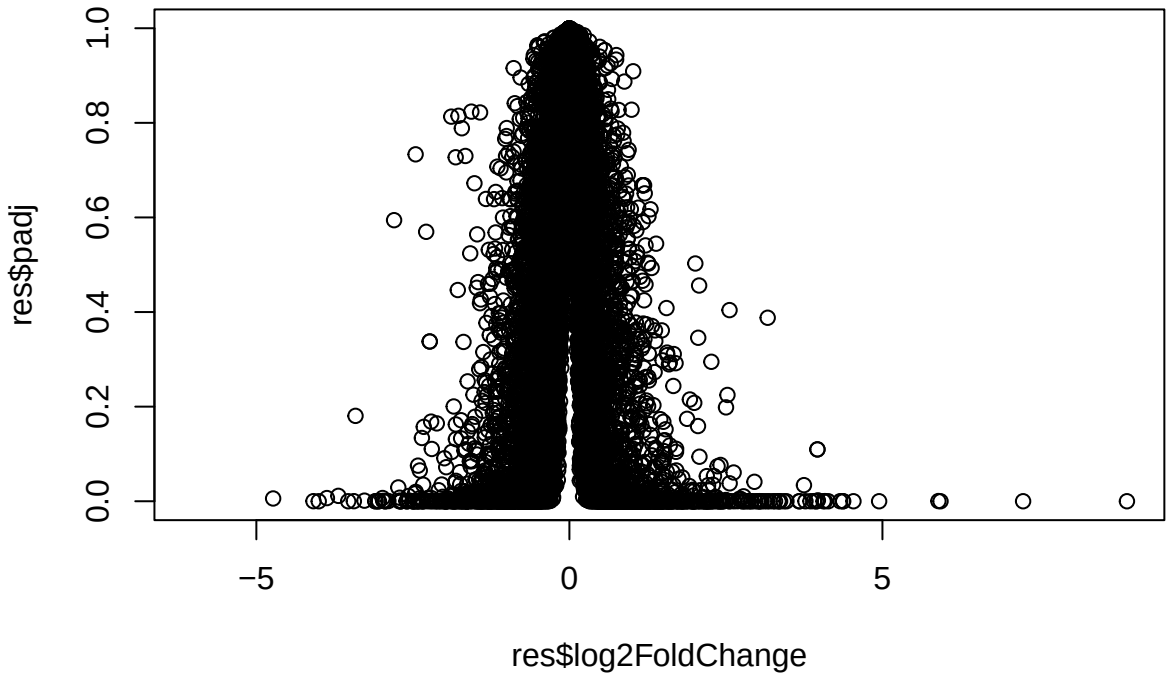
```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

```
#View results
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
```

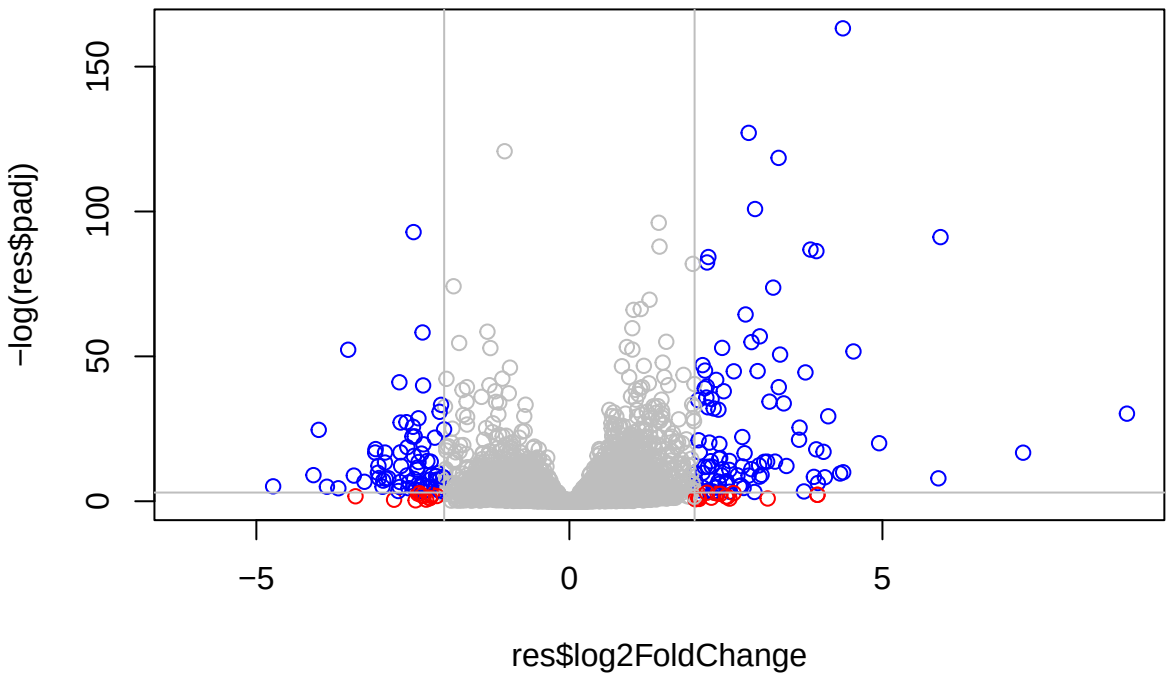
Volcano Plots

```
#Raw DESeq2 output plot
plot(res$log2FoldChange,res$padj)
```




```
#Make custom color vector for plot

##Make vector with default color, same length as the number of plotted pts
custom.color <- rep("gray", nrow(res))

##Overwrite high fold change points with red
custom.color[abs(res$log2FoldChange) > 2] <- "red"

##Overwrite points with high p-value with blue
custom.color[(res$padj < 0.05) & (abs(res$log2FoldChange) > 2)] <- "blue"
###High p-value = small -log(padj), since the negative of the log will flip these paints from near zero
```

```
#Log DESeq2 output plot & annotate by color
plot(res$log2FoldChange,-log(res$padj),
     col=custom.color)
abline(h=-log(0.05), col="gray")
abline(v=c(-2,2), col="gray")
```



Significant results are the blue dots. The blue dots are both higher than 2-fold increase in either direction, and have p<0.05, making these results significant.