BIMM 143 Class 13

Xaler Lu (A17388454)

Background

We will use published RNA-Seq data to analyze the effects of a common steroid on airway cells. Dexamethasone (dex) is a synthetic steroid that reduces asthma inflammation of the airways.

Data Import

```
library(BiocManager)
```

```
Warning: package 'BiocManager' was built under R version 4.3.3
```

```
library(DESeq2)
```

```
Warning: package 'DESeq2' was built under R version 4.3.3
```

```
Warning: package 'S4Vectors' was built under R version 4.3.2
```

```
Warning: package 'GenomeInfoDb' was built under R version 4.3.3
```

```
Warning: package 'SummarizedExperiment' was built under R version 4.3.2
```

```
Warning: package 'matrixStats' was built under R version 4.3.3
```

We need two different inputs:
- `countData`
- `colData`: Meta data that describes the colums in countData

```
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
```

```
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
```

Q1. How many genes are in this dataset?

This dataset has 38694 genes

```
nrow(counts)
```

```
[1] 38694
```

Q2 How many "control" cell lines do we have?

This dataset has 4 control cell lines.

```
table(metadata$dex)
```

```
control treated
      4       4
```

Differential Gene Expression

We have four replicate drug treated and control columns in our `counts`. We want a "mean" value for each genes (rows) in "treated" and one mean value for each gene in "control".

The code below filters out all the control columns and treated columns in the `counts` data. Then, it calculates the mean either using `rowSums()` and divide by the number of controls, or using `rowMeans()` by itself.

```
library(dplyr)
```

```
control <- metadata %>% filter(dex == "control")
control.counts <- counts %>% select(control$id)
control.mean <- rowSums(control.counts)/4
```

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

The names of the samples would throw off our filtering because it may not match the names "control" or "treated".

With a changing amount of samples, then `rowSums()` won't work by itself, so `rowMeans()` would suit the code much better.

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 <- metadata %>% filter(dex == "treated")
treated.counts <- counts %>% select(treated$id)
treated.mean <- rowMeans(treated.counts)
```

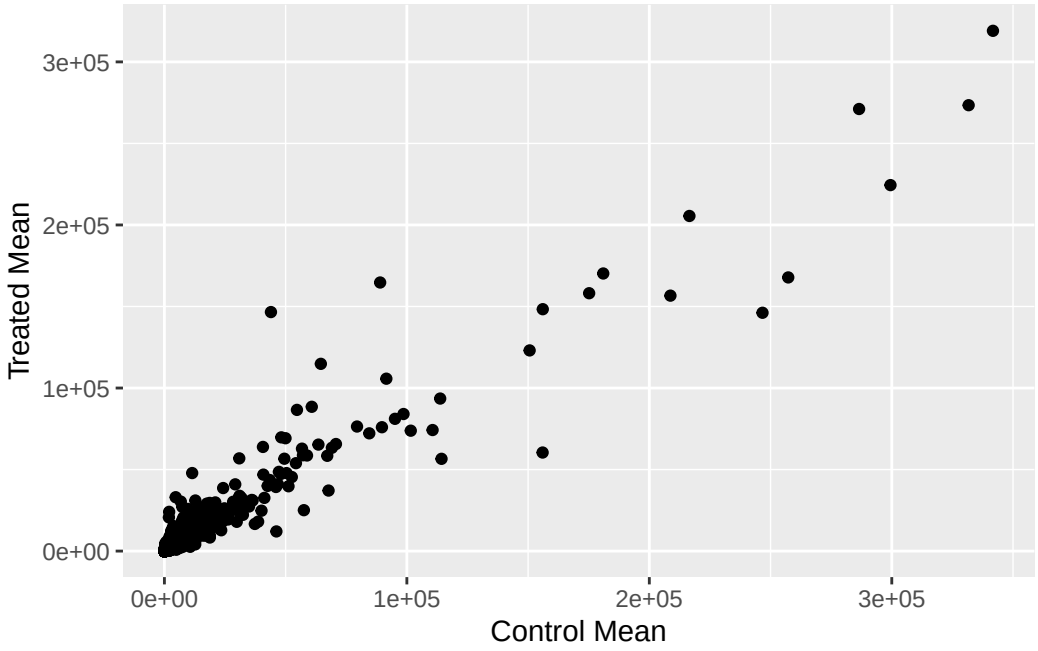
Q5a. Create a scatter plot showing the mean of the treated samples against the mean of the control samples. Your plot should look something like the following.

```
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 4.3.3
```

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

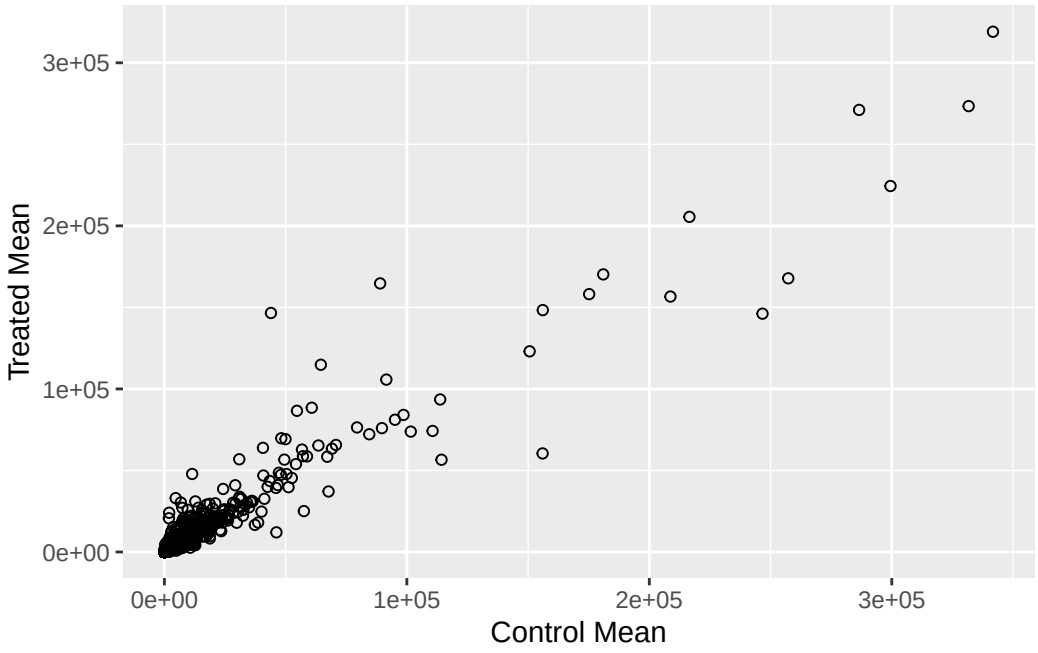
```
ggplot(mean.all) +
  aes(control.mean, treated.mean) +
  geom_point() +
  labs(x = "Control Mean", y= "Treated Mean")
```



Q5b. 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(shape = 1)` gives us open circles.

```
ggplot(mean.all) +
  aes(control.mean, treated.mean) +
  geom_point(shape = 1) +
  labs(x = "Control Mean", y= "Treated Mean")
```



Let's log transform the data

N.B. We often use log2 for this type of data.

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

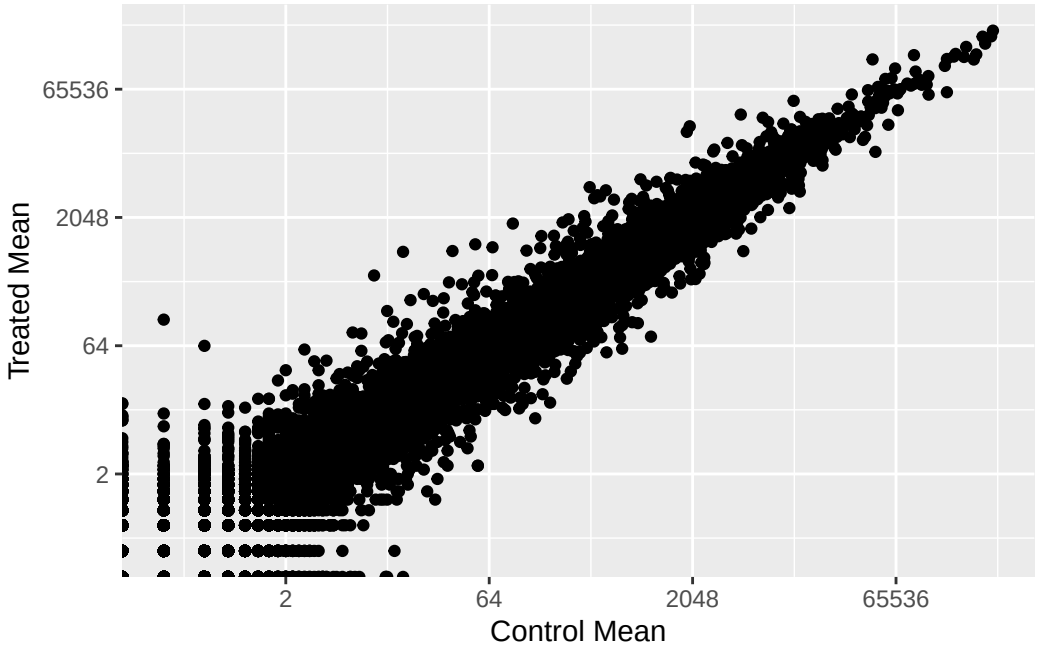
We can use `plot(data, log2 = "xy")`

```
ggplot(mean.all) +
  aes(control.mean, treated.mean) +
  geom_point() +
  scale_x_continuous(trans = 'log2') +
  scale_y_continuous(trans = 'log2') +
  labs(x = "Control Mean", y= "Treated Mean")
```

```
Warning in scale_x_continuous(trans = "log2"): log-2 transformation introduced
```

```
infinite values.

Warning in scale_y_continuous(trans = "log2"): log-2 transformation introduced
infinite values.
```




```
plot(mean.all, log2 = "xy")
```

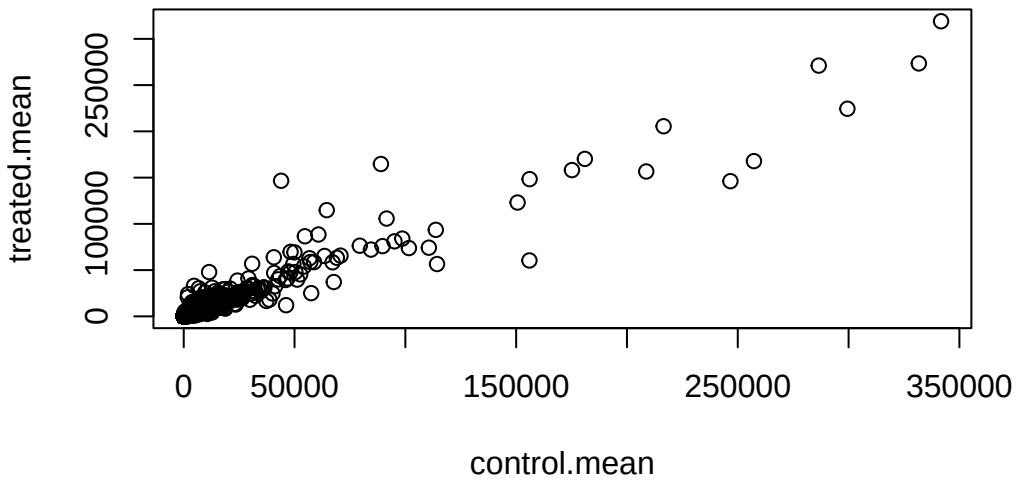
```
Warning in plot.window(...): "log2" is not a graphical parameter

Warning in plot.xy(xy, type, ...): "log2" is not a graphical parameter

Warning in axis(side = side, at = at, labels = labels, ...): "log2" is not a
graphical parameter
Warning in axis(side = side, at = at, labels = labels, ...): "log2" is not a
graphical parameter

Warning in box(...): "log2" is not a graphical parameter

Warning in title(...): "log2" is not a graphical parameter
```



Treated/Control is often called **folded change**

We want to add this newly calculated `folded change` as a new column onto `mean.all` called `log2fc`.

```
mean.all$log2fc <- log2( mean.all$treated.mean/mean.all$control.mean)
head(mean.all)
```

```
                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
```

When we log transform, we get `NaN` and `-Inf` when we divide by zero or take the log of zero, respectively. Thus, we want to filter these data out.

```
zero.vals <- which(mean.all[,1:2]==0, arr.ind=TRUE)

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

```
mycounts <- mean.all[-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
```

> 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 both row and column positions with TRUE values. In this case, we are looking at columns with zero counts.

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

We want to see fold change less than -2 or greater than 2. We can sum up the up-regulated gens and down-regulated genes with `sum()`

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

```
sum(up.ind)
```

```
[1] 250
```

> 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(down.ind)
```

```
[1] 367
```

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

We haven't done a statistical tests and some of these may not be statistically significant.

DESeq Analysis

Let's do this analysis with an estimate of statistical signifiance using the **DESeq2** package. Remember to load in the library.

DESeq, like many bioconductor packages, wants its input data in a very specific way.

We want to use the function `DESeqDataSetFromMatrix()` with three required arguments: `countData`, `colData`, and `design`.

```
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
```

Running the DESeq Analysis Pipeline

The main function `DESeq()` takes the data from the dataset from matrix `dds`

```
dds <- DESeq(dds)
```

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

To get the results, use `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
```

In the results, the `padj` column is the adjusted value of p-values. We use `padj` because we have 36000 genes and the alpha level of 0.05 will yield around 1800 genes that are false positive. This means `padj` is more stringent than p-values.

Adding Annotation Data

Our table uses Ensembl gene IDs, but alternative gene names and extra annotations are required for interpretations.

```
library("AnnotationDbi")
```

```
Attaching package: 'AnnotationDbi'
```

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

    select
```

```
library("org.Hs.eg.db")
```

```
columns(org.Hs.eg.db)
```

```
 [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
 [6] "ENTREZID"     "ENZYME"       "EVIDENCE"     "EVIDENCEALL"  "GENENAME"
[11] "GENETYPE"     "GO"           "GOALL"        "IPI"          "MAP"
[16] "OMIM"         "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"
[21] "PMID"         "PROSITE"      "REFSEQ"       "SYMBOL"       "UCSCKG"
[26] "UNIPROT"
```

```
res$symbol <- mapIds(org.Hs.eg.db,
                     keys=row.names(res), # Our genenames
                     keytype="ENSEMBL",        # The format of our genenames
                     column="SYMBOL",          # The new format we want to add
                     multiVals="first")
```

```
'select()' returned 1:many mapping between keys and columns
```

```
head(res)
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 7 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      symbol
                <numeric> <character>
ENSG00000000003  0.163035      TSPAN6
ENSG00000000005        NA        TNMD
ENSG00000000419  0.176032        DPM1
ENSG00000000457  0.961694       SCYL3
ENSG00000000460  0.815849       FIRRM
ENSG00000000938        NA         FGR
```

Q11. Run the mapIds() function two more times to add the Entrez ID and UniProt accession and GENENAME as new columns called res$entrez, res$uniprot and res$genename.

```
res$entrez <- mapIds(org.Hs.eg.db,
                     keys=row.names(res),
                     column="ENTREZID",
                     keytype="ENSEMBL",
                     multiVals="first")
```

```
'select()' returned 1:many mapping between keys and columns
```

```
res$uniprot <- mapIds(org.Hs.eg.db,
                     keys=row.names(res),
                     column="UNIPROT",
                     keytype="ENSEMBL",
                     multiVals="first")
```

```
'select()' returned 1:many mapping between keys and columns
```

```
res$genename <- mapIds(org.Hs.eg.db,
                     keys=row.names(res),
                     column="GENENAME",
                     keytype="ENSEMBL",
                     multiVals="first")
```

```
'select()' returned 1:many mapping between keys and columns
```

```
head(res)
```

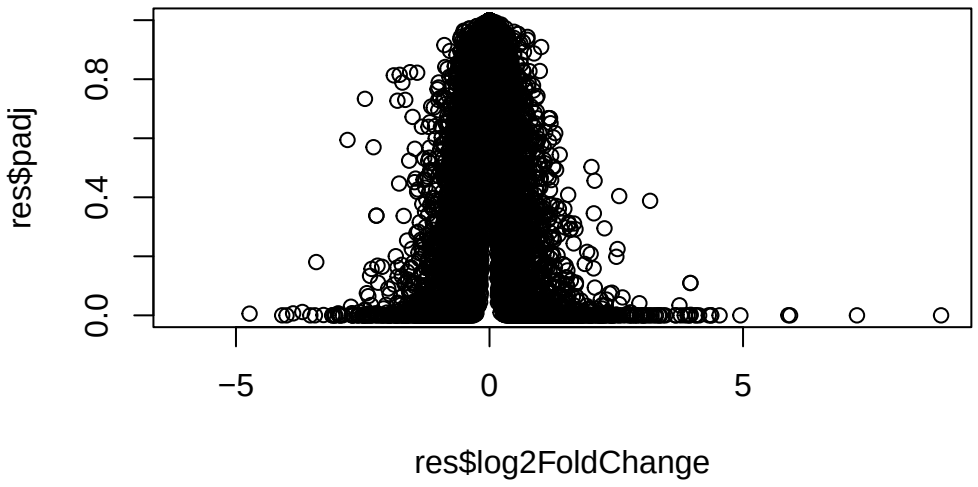
```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 10 columns
                 baseMean log2FoldChange     lfcSE      stat    pvalue
                <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195     -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      symbol      entrez     uniprot
                <numeric> <character> <character> <character>
ENSG00000000003  0.163035      TSPAN6        7105  A0A024RCI0
ENSG00000000005        NA        TNMD       64102      Q9H2S6
ENSG00000000419  0.176032        DPM1        8813      O60762
ENSG00000000457  0.961694       SCYL3       57147      Q8IZE3
ENSG00000000460  0.815849       FIRRM       55732  A0A024R922
ENSG00000000938        NA         FGR        2268      P09769
                          genename
                       <character>
ENSG00000000003          tetraspanin 6
ENSG00000000005            tenomodulin
ENSG00000000419 dolichyl-phosphate m..
ENSG00000000457 SCY1 like pseudokina..
ENSG00000000460 FIGNL1 interacting r..
ENSG00000000938 FGR proto-oncogene, ..
```

Volcano Plot

This is a main summary of the results from these kinds of studies.

We want small `padj` values, but a majority are above the alpha level. We want to log transform the `padj` values

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



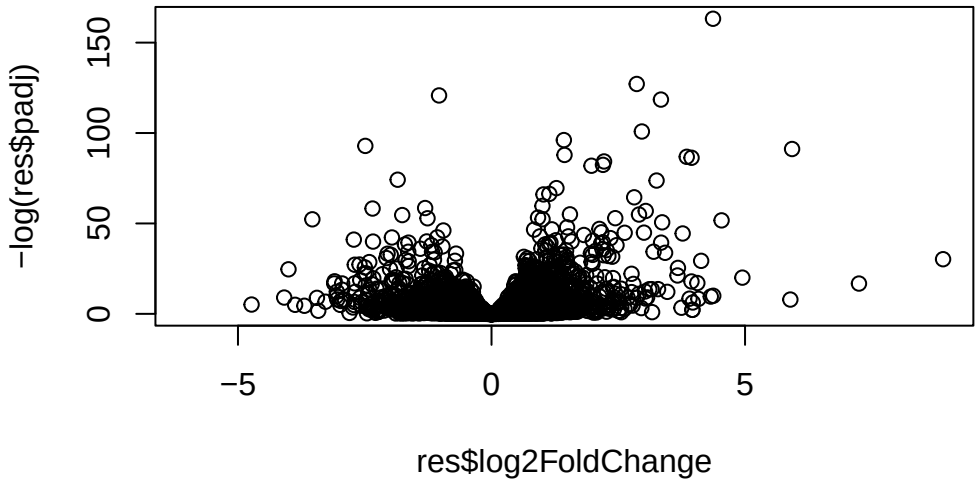

```
ord <- order( res$padj )
#View(res[ord,])
head(res[ord,])
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 10 columns
                 baseMean log2FoldChange     lfcSE      stat      pvalue
                <numeric>      <numeric> <numeric> <numeric>   <numeric>
ENSG00000152583   954.771        4.36836 0.2371268   18.4220 8.74490e-76
ENSG00000179094   743.253        2.86389 0.1755693   16.3120 8.10784e-60
ENSG00000116584  2277.913       -1.03470 0.0650984  -15.8944 6.92855e-57
ENSG00000189221  2383.754        3.34154 0.2124058   15.7319 9.14433e-56
ENSG00000120129  3440.704        2.96521 0.2036951   14.5571 5.26424e-48
ENSG00000148175 13493.920        1.42717 0.1003890   14.2164 7.25128e-46
                      padj      symbol      entrez     uniprot
```

```
                  <numeric> <character> <character> <character>
ENSG00000152583 1.32441e-71     SPARCL1        8404  A0A024RDE1
ENSG00000179094 6.13966e-56        PER1        5187      O15534
ENSG00000116584 3.49776e-53     ARHGEF2        9181      Q92974
ENSG00000189221 3.46227e-52        MAOA        4128      P21397
ENSG00000120129 1.59454e-44       DUSP1        1843      B4DU40
ENSG00000148175 1.83034e-42        STOM        2040      F8VSL7
                            genename
                         <character>
ENSG00000152583           SPARC like 1
ENSG00000179094 period circadian reg..
ENSG00000116584 Rho/Rac guanine nucl..
ENSG00000189221    monoamine oxidase A
ENSG00000120129 dual specificity pho..
ENSG00000148175               stomatin
```

For the next version, the log transformation turns the values negative, so we want to turn the values positive.

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

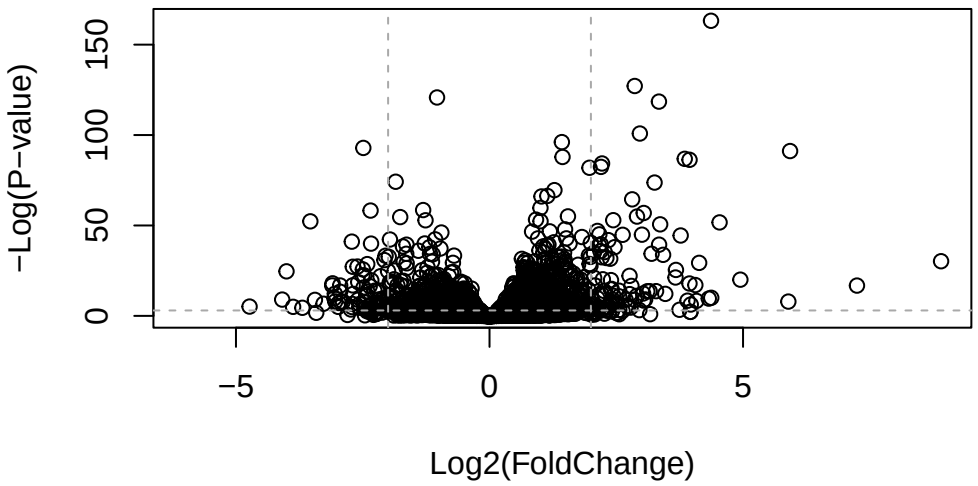


Let's add some guidelines. The points above the horizontal line are statistically significant

because of the negative log transformation. The points outside the vertical lines (>2 or <-2 folded change) show substantial differences between treatment and control.

```
plot( res$log2FoldChange,  -log(res$padj),
 ylab="-Log(P-value)", xlab="Log2(FoldChange)")

abline(v=c(-2,2), col="darkgray", lty=2)
abline(h=-log(0.05), col="darkgray", lty=2)
```



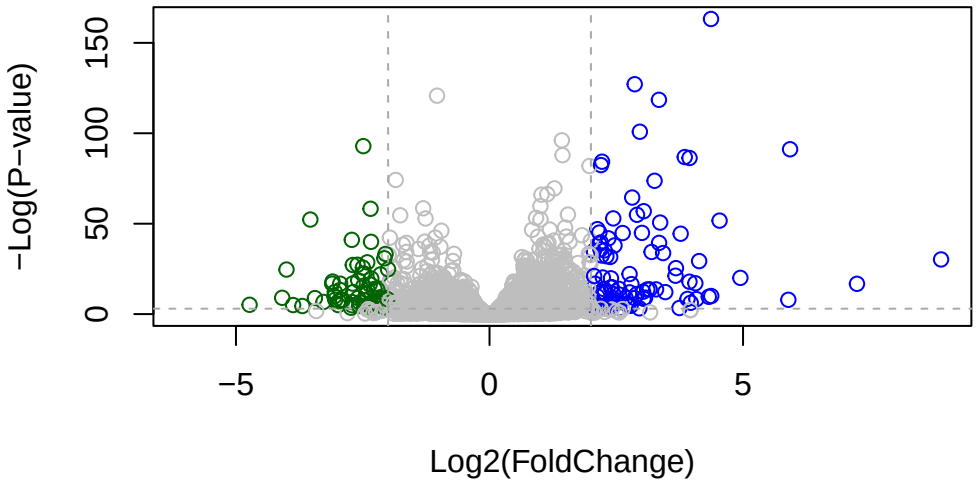
To better visualize this, let's add color to the points that matter.

```
mycols <- rep("gray", nrow(res))
mycols[res$log2FoldChange > 2] <- "blue"
mycols[res$log2FoldChange < -2] <- "darkgreen"
mycols[res$padj >= 0.05] <- "gray"


plot( res$log2FoldChange,  -log(res$padj), col = mycols,
 ylab="-Log(P-value)", xlab="Log2(FoldChange)")

abline(v=c(-2,2), col="darkgray", lty=2)
```

```
abline(h=-log(0.05), col="darkgray", lty=2)
```

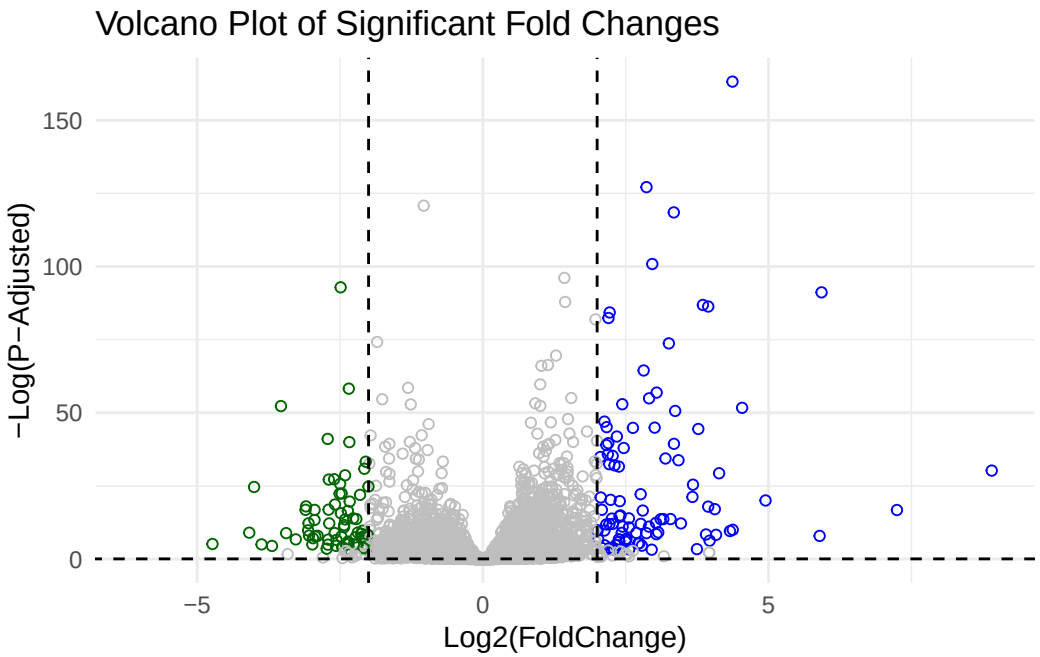



```
library(tidyr)
```

Here is the ggplot version. Assign the colors to `geom_point()`, not `aes()`.

```
ggplot(res) +
  aes(log2FoldChange, -log(padj)) +
  geom_point(shape = 1, col= mycols) +
  geom_abline(intercept = 0.05, slope = 0, lty = 2) +
  geom_vline(xintercept = c(-2,2), lty = 2) +
  labs(title = "Volcano Plot of Significant Fold Changes", x = "Log2(FoldChange)", y= "-Lo
  theme_minimal()
```

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



Saving Our Results

Write a CSV file

```
write.csv(res, file = "class13results.csv")
```