

class 13 Transcriptomics and the analysis of RNA-Seq data

Trinity Lee A16639698

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).

##2. Bioconductor setup

```
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, 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
```

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

```
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
```

```
##3. Import countData and colData
```

```
# Complete the missing code
counts <- read.csv("airway_scaledcounts.csv", row.names=1)
metadata <- read.csv("airway_metadata.csv")
head(counts)
```

	SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516
ENSG000000000003	723	486	904	445	1170
ENSG000000000005	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
ENSG000000000003	1097	806	604
ENSG000000000005	0	0	0
ENSG000000000419	781	417	509
ENSG000000000457	447	330	324
ENSG000000000460	94	102	74
ENSG000000000938	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

Q1. How many genes are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

Q2. How many ‘control’ cell lines do we have?

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

control	treated
4	4

##4. Toy differential gene expression Let’s start by calculating the gene mean counts for “control” samples and compare to the mean counts per gene for the “treated” values.

Let’s start with the finding which columns in `counts` are control samples then calculate the mean value:

```
control <- metadata[metadata[, "dex"] == "control",]
control.counts <- counts[, control$id]
control.mean <- rowSums(control.counts) / 4
head(control.mean)
```

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

An alternative way to do this same thing using the `dplyr` package from the tidyverse is shown below. Which do you prefer and why?

```
library(dplyr)
```

```
Attaching package: 'dplyr'
```

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

```
combine
```

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

```
count
```

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

```
intersect, setdiff, union
```

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

```
intersect
```

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

```
collapse, desc, intersect, setdiff, slice, union
```

```
The following objects are masked from 'package:S4Vectors':
```

```
  first, intersect, rename, setdiff, setequal, union
```

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

```
  combine, intersect, setdiff, union
```

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

```
  filter, lag
```

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

```
  intersect, setdiff, setequal, union
```

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

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

I prefer not using the dplyr package since the library does not have to be loaded before use and less typing is used in order to accomplish the same task.

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

using `rowMeans` function instead of `rowSums`

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

```
ENSG000000000003 ENSG000000000005 ENSG000000000419 ENSG000000000457 ENSG000000000460
         900.75          0.00        520.50        339.75        97.25
ENSG000000000938
         0.75
```

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

```
ENSG000000000003 ENSG000000000005 ENSG000000000419 ENSG000000000457 ENSG000000000460
       658.00          0.00        546.00        316.50        78.75
ENSG000000000938
       0.00
```

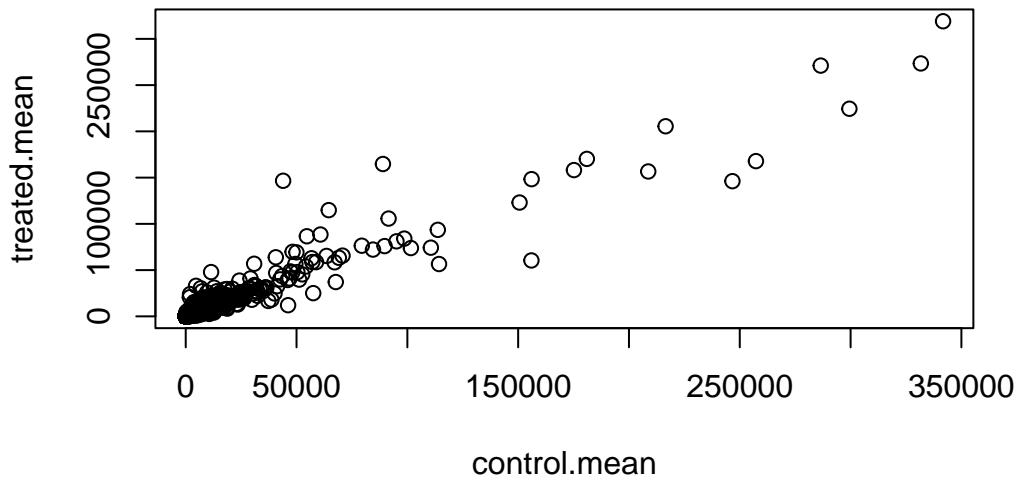
combine meancount data and show the sum of the mean counts for all genes for each group

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

control.mean treated.mean
23005324      22196524
```

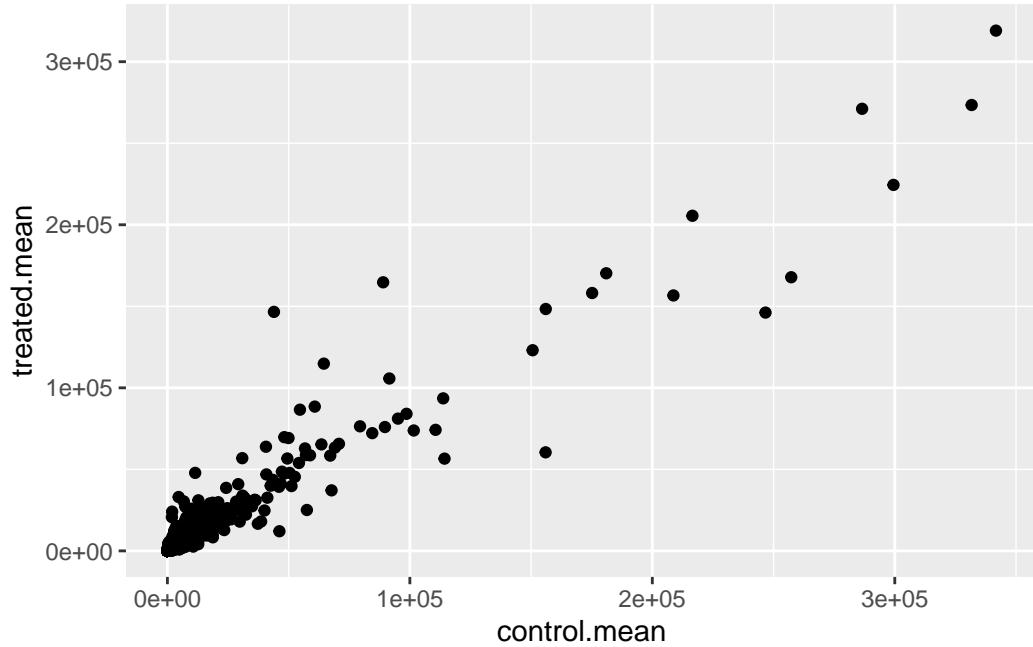
Q5 (a). Create a scatter plot showing the mean of the treated samples against the mean of the control samples.

```
plot(meancounts)
```



Q5 (b). You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot?

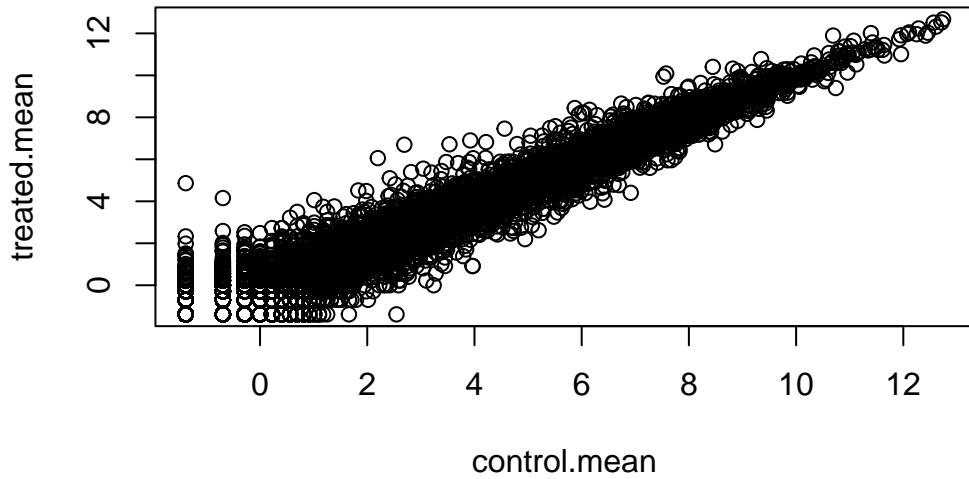
```
library(ggplot2)
ggplot(meancounts)+aes(control.mean,treated.mean)+geom_point()
```



Wait a sec. There are 60,000-some rows in this data, but I'm only seeing a few dozen dots at most outside of the big clump around the origin.

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

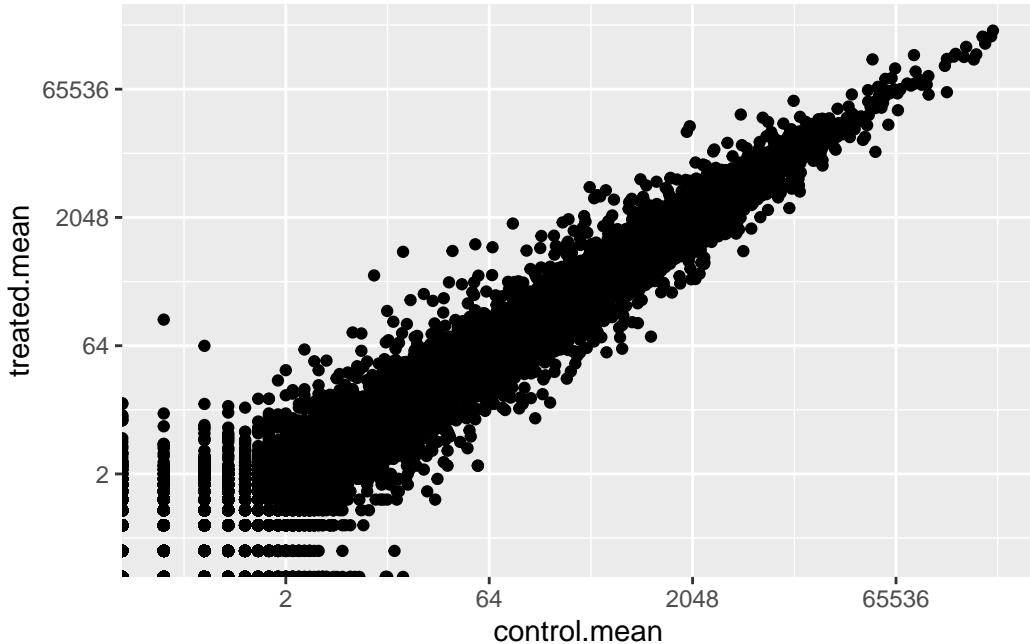
```
plot(log(meancounts))
```



```
ggplot(meancounts)+aes(control.mean,treated.mean)+geom_point()+scale_x_continuous(trans="log")
```

Warning: Transformation introduced infinite values in continuous x-axis

Warning: Transformation introduced infinite values in continuous y-axis



Log transformation is useful when data is skewed and measured over a wide range. Log 2 is the most preferred.

Here we calculate log2foldchange, add it to our meancounts data.frame and inspect the results either with the head() or the View() function for example.

```
meancounts$log2fc <- log2(meancounts[, "treated.mean"] / meancounts[, "control.mean"])
head(meancounts)
```

	control.mean	treated.mean	log2fc
ENSG000000000003	900.75	658.00	-0.45303916
ENSG000000000005	0.00	0.00	NaN
ENSG000000000419	520.50	546.00	0.06900279
ENSG000000000457	339.75	316.50	-0.10226805
ENSG000000000460	97.25	78.75	-0.30441833
ENSG000000000938	0.75	0.00	-Inf

Let's filter our data to remove genes with zero expression to get rid of "NaN" and "-Inf" results

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

```
mycounts <- meancounts[-to.rm,]  
head(mycounts)
```

	control.mean	treated.mean	log2fc
ENSG000000000003	900.75	658.00	-0.45303916
ENSG000000000419	520.50	546.00	0.06900279
ENSG000000000457	339.75	316.50	-0.10226805
ENSG000000000460	97.25	78.75	-0.30441833
ENSG000000000971	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?

The **arr.ind** function allows us to identify the genes and samples that have zero counts so that we can ignore them and focus on the **row** answer to get rid of any “NaN” or “-Inf” results. The **unique()** function is called to make sure that the row is not counted twice if there is zero for both samples.

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

```
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(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?

no because we do not know how to compare these results to know whether the difference is statistically significant.

##5. Setting up for DESeq like any package we must load it with library()

```
library(DESeq2)
citation("DESeq2")
```

To cite package 'DESeq2' in publications use:

Love, M.I., Huber, W., Anders, S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2 Genome Biology 15(12):550 (2014)

A BibTeX entry for LaTeX users is

```
@Article{,
  title = {Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2},
  author = {Michael I. Love and Wolfgang Huber and Simon Anders},
  year = {2014},
  journal = {Genome Biology},
  doi = {10.1186/s13059-014-0550-8},
  volume = {15},
  issue = {12},
  pages = {550},
}
```

```
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)
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>
ENSG000000000003 747.194195 -0.3507030 0.168246 -2.084470 0.0371175
ENSG000000000005 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>
ENSG000000000003 0.163035
ENSG000000000005      NA
ENSG00000000419 0.176032
ENSG00000000457 0.961694
ENSG00000000460 0.815849
ENSG00000000938      NA

```

A summary results plot

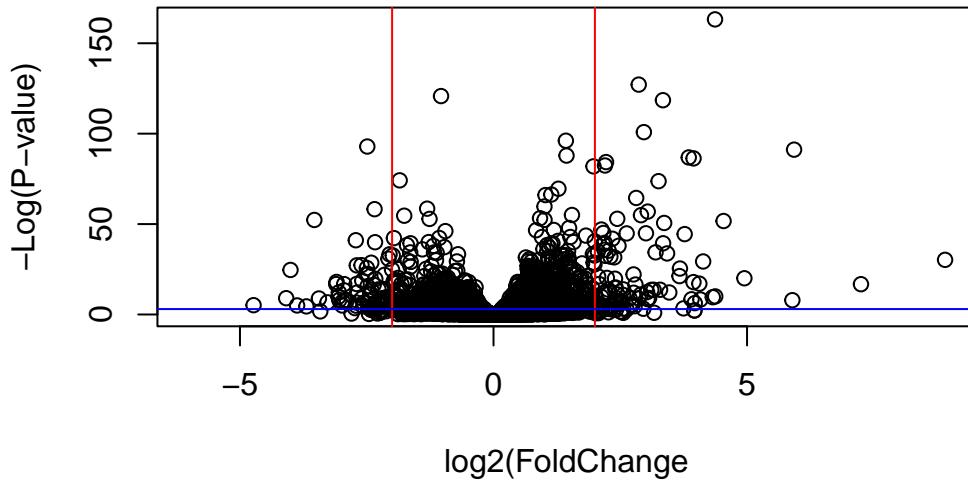
Volcano plot.

This is a plot that is a common type of summary figure that shows both P-values and Log2(fold-changes).

```

plot(res$log2FoldChange,-log(res$padj),xlab="log2(FoldChange",ylab="-Log(P-value)")
abline(v=2,col="red")
abline(v=-2,col="red")
abline(h=-log(0.05),col="blue")

```



Create a nicer plot using ggplot

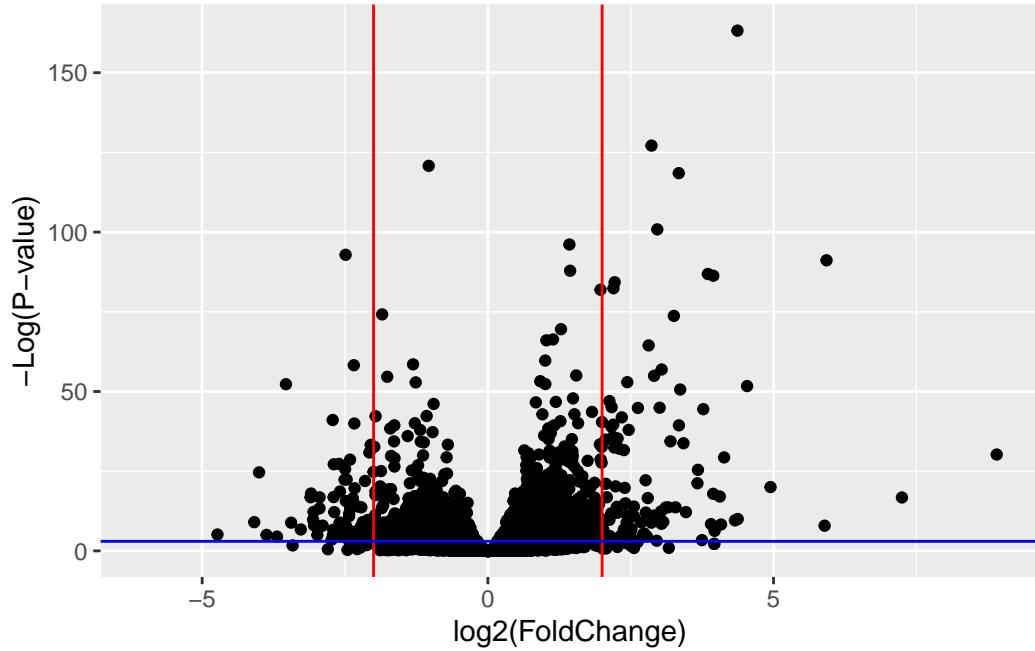
```

# Extract results as a data frame
results_df <- as.data.frame(res)

# Create a ggplot
ggplot(results_df, aes(x = log2FoldChange, y = -log(padj))) +
  geom_point() +
  labs(x = "log2(FoldChange)", y = "-Log(P-value)") +
  geom_vline(xintercept = c(-2, 2), color = "red") +
  geom_hline(yintercept = -log(0.05), color = "blue")

```

Warning: Removed 23549 rows containing missing values (`geom_point()`).



Save our results to date ...

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