

Class 12: Transcriptomics and the analysis of RNA-Seq data

Challana Tea

```
library(BiocManager)
library(ggplot2)
```

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
ENSG000000000003	723	486	904	445	1170
ENSG000000000005	0	0	0	0	0
ENSG000000000419	467	523	616	371	582
ENSG000000000457	347	258	364	237	318
ENSG000000000460	96	81	73	66	118
ENSG000000000938	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		

```
nrow(counts)
```

```
[1] 38694
```

Q1. How many genes are in this dataset?

38694 genes

```
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
```

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

We have 4 control cell lines

Toy differential gene expression

Find the sample id for those labeled control.

```
control <- metadata[metadata[, "dex"] == "control", ]
control$id
```

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

Then calculate the mean counts per gene across these samples:

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

```

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

```

Q3. How would you make the above code in more robust?

Adjust mean calculation such that the sums are divided by the number of rows labeled control instead just 4

```

control <- metadata[metadata[, "dex"]=="control",]
control$id

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

control.counts <- counts[ ,control$id]
control.mean <- rowSums(control.counts)/nrow(control)
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 <- rowSums(counts[ ,treated$id])/nrow(control)
head(treated.mean)

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

```

Store the two results into a data frame

```

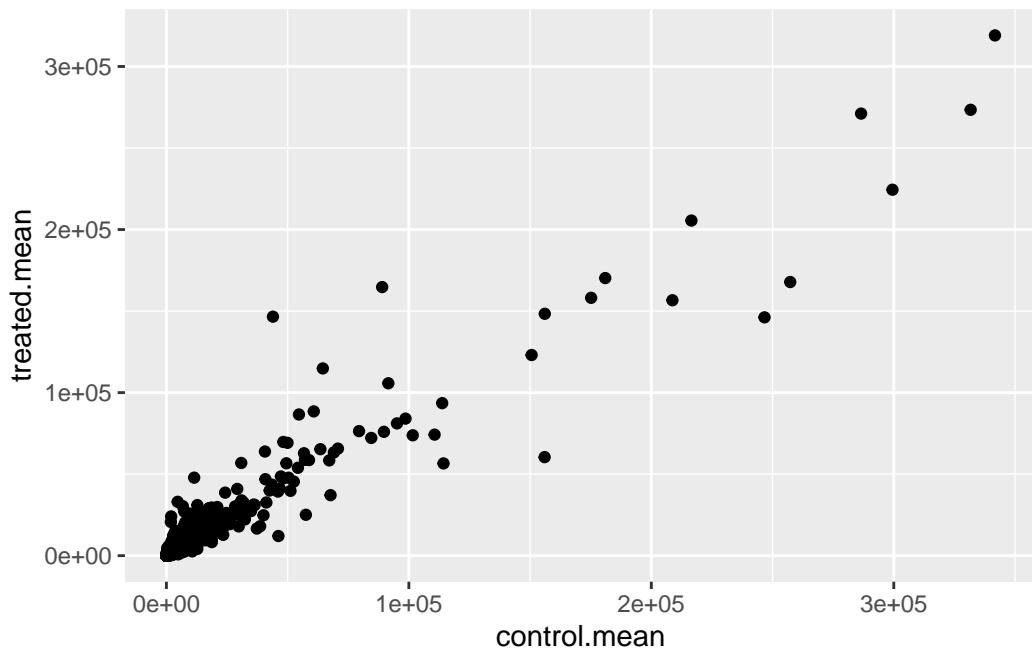
meancounts <- data.frame(control.mean, treated.mean)
head(meancounts)

```

	control.mean	treated.mean
ENSG000000000003	900.75	658.00
ENSG000000000005	0.00	0.00
ENSG00000000419	520.50	546.00
ENSG00000000457	339.75	316.50
ENSG00000000460	97.25	78.75
ENSG00000000938	0.75	0.00

Q5. 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.

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



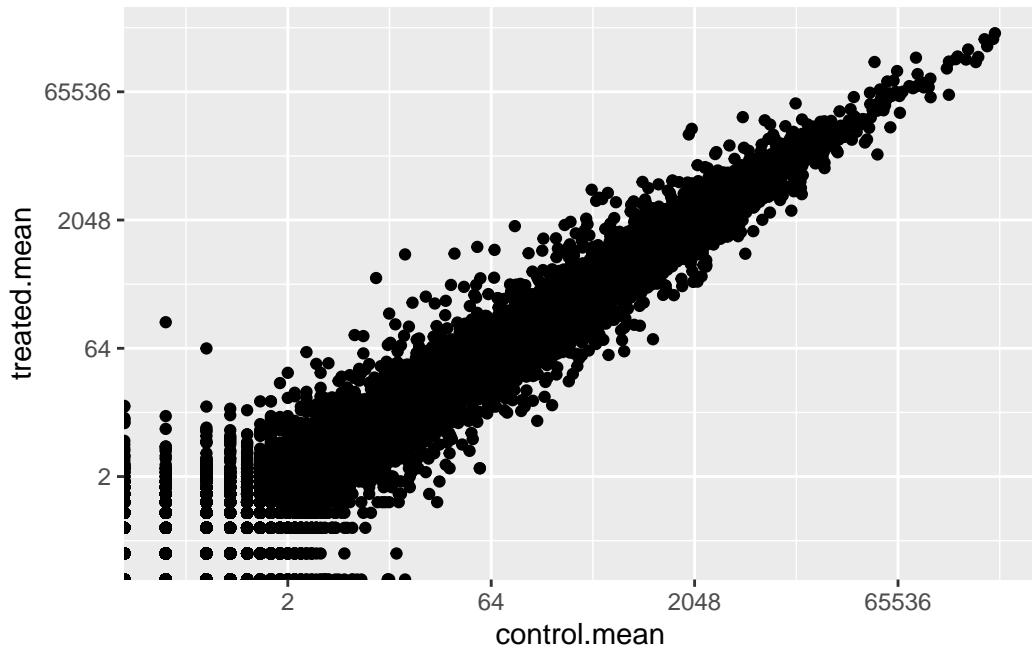
Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

```
ggplot(meancounts) +
  aes(control.mean, treated.mean) +
  scale_x_continuous(trans="log2") +
  scale_y_continuous(trans="log2") +
```

```
geom_point()
```

Warning: Transformation introduced infinite values in continuous x-axis

Warning: Transformation introduced infinite values in continuous y-axis



Calculate the log2 Fold Change for treated and control mean counts

```
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
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

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

```
[1] NA
```

Filter data to remove genes with zero expression

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

	row	col
ENSG000000000005	2	1
ENSG00000004848	65	1
ENSG00000004948	70	1
ENSG00000005001	73	1
ENSG00000006059	121	1
ENSG00000006071	123	1

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

	control.mean	treated.mean	log2fc
ENSG000000000003	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 specifies exactly where the zeroes are. unique() is called to make sure the same row isn't called twice if it has two zeroes.

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

Q8 & 9. Using the up.ind vector above can you determine how many up regulated genes we have at the greater than 2 fc level? Using the down.ind vector above can you determine how many down regulated genes we have at the greater than 2 fc level?

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

```
[1] 250
```

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

```
[1] 367
```

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

There's no statistical analysis performed deciding whether the fold-change is 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 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
```

```
Import data
```

```
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

Retrieve results with DESeq function `results()`

```
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

```

```

res05 <- results(dds, alpha=0.05)
summary(res05)

```

```

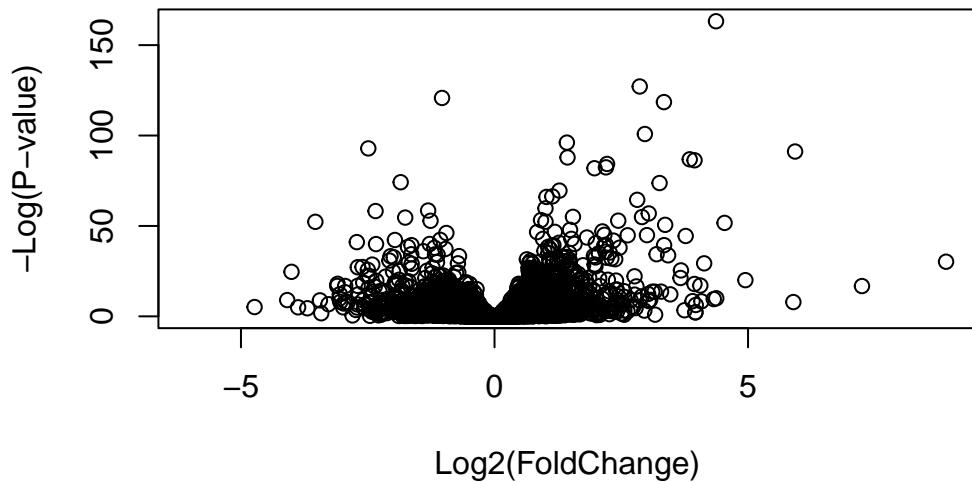
out of 25258 with nonzero total read count
adjusted p-value < 0.05
LFC > 0 (up)      : 1236, 4.9%
LFC < 0 (down)     : 933, 3.7%
outliers [1]       : 142, 0.56%
low counts [2]      : 9033, 36%
(mean count < 6)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results

```

Data Visualization

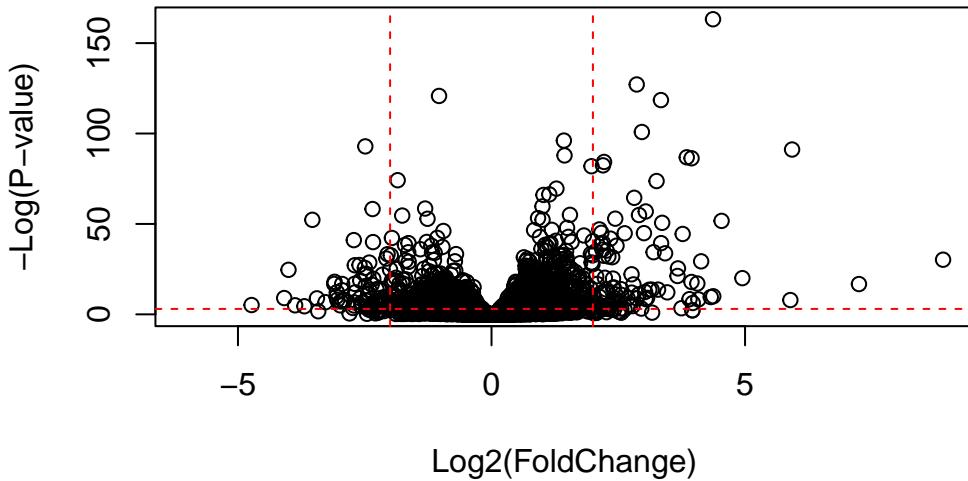
Plot on base R a volcano plot

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



Add cut-off lines to highlight genes with a padj < 0.05 and an absolute logFC > 2

```
plot( res$log2FoldChange, -log(res$padj),  
      ylab="-Log(P-value)", xlab="Log2(FoldChange)")  
  
# Cut-off lines  
abline(v=c(-2,2), col="red", lty=2)  
abline(h=-log(0.05), col="red", lty=2)
```



Adjust the plot to indicate transcripts with high fold-change and significant difference

```

# Setup our custom point color vector
mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ] <- "yellow"

inds <- (res$padj < 0.01) & (abs(res$log2FoldChange) > 2 )
mycols[ inds ] <- "blue"

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

# Cut-off lines
abline(v=c(-2,2), col="red", lty=2)
abline(h=-log(0.1), col="red", lty=2)

```

