Class 13 DESeq

Ava Limtiaco (PID:A18007672)

2025-05-13

This week we are looking at differential expression analysis.

This 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).

#Import the data from Himes et al.

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

Let's look at the data from each 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
```

Sanity check on correspondenace of counts and metadata.

```
all(metadata$id == colnames(counts))
```

```
[1] TRUE
```

```
all(c(T,T,F,T))
```

```
[1] FALSE
```

Q1. How many genes are in this dataset?

There are 38694 genes in this dataset.

```
#nrow(mycounts)
```

Q2. How many 'control' cell lines do we have?

```
n.control <- sum(metadata$dex == "control")
```

There are 4 control cell lines.

###Extract and summarize control samples.

To find out where the control samples are we need the metadata.

```
counts <- read.csv("airway_scaledcounts.csv")
```

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

```
[1] 900.75   0.00 520.50 339.75  97.25   0.75
```

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

You could use library(dplyr) instead.

```
library(dplyr)
```

```
Attaching package: 'dplyr'

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)
```

```
[1] 900.75   0.00 520.50 339.75  97.25   0.75
```

Extract and summarize the treat samples.

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

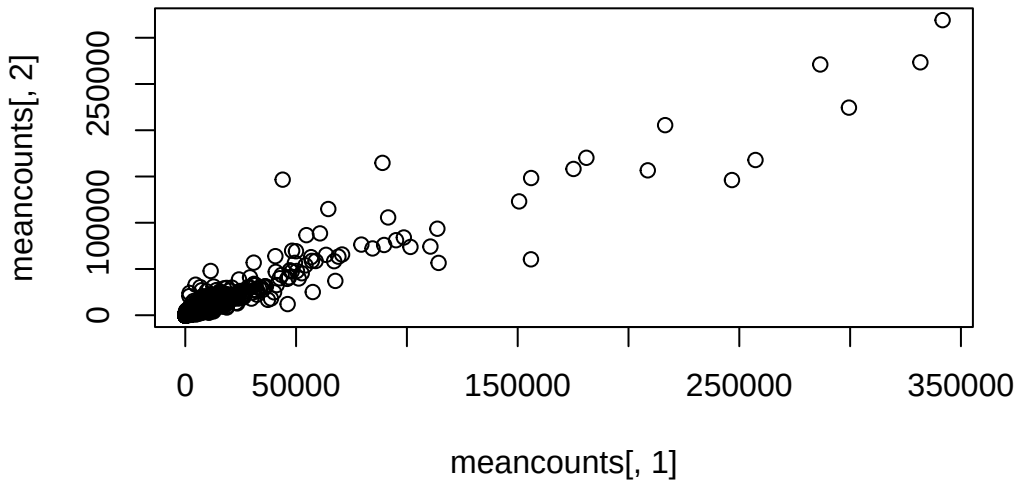
```
[1] 658.00   0.00 546.00 316.50  78.75   0.00
```

Store these results together in a new data frame called 'meancounts'.

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

Let's make a plot to explore these results a little.

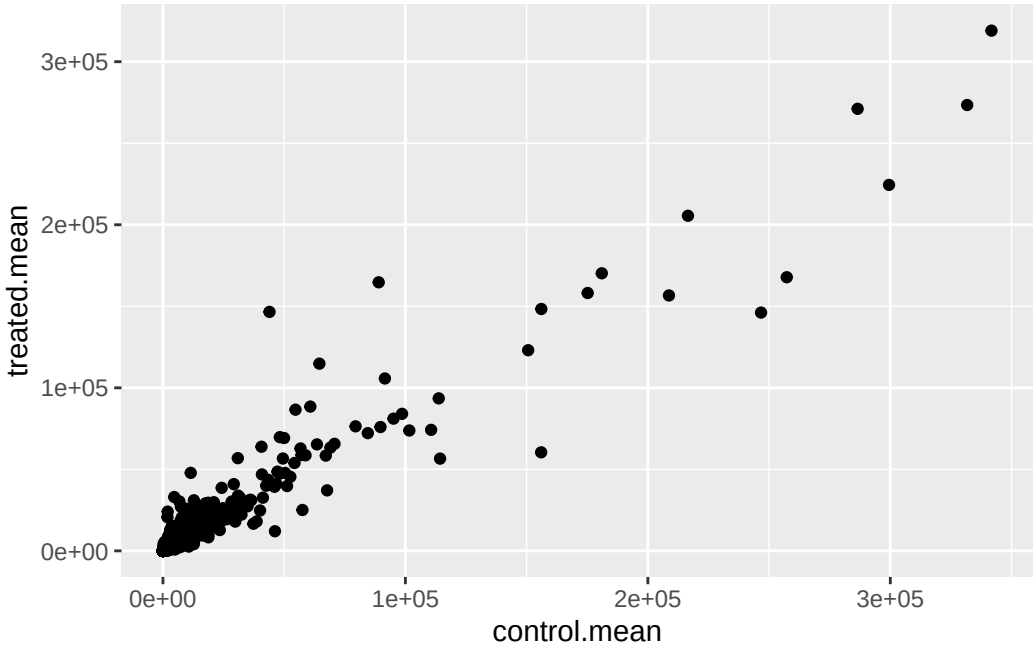
```
plot(meancounts[,1], meancounts[,2])
```



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

You would use geom_point().

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



We will make a log-log plot to draw out this skewed data and see what is going on.

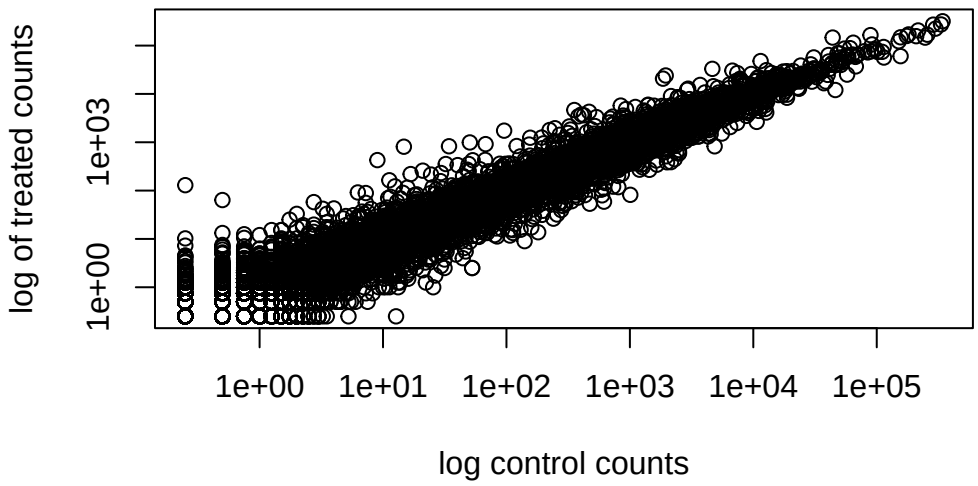
> Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

This is the function:

```
plot(meancounts[,1], meancounts[,2], log="xy", xlab="log control counts", ylab="log of treate
```

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

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



We often log2 transformations when dealing with this sort of data.

```
log2(20/20)
```

```
[1] 0
```

```
log2(40/20)
```

```
[1] 1
```

```
log2(20/40)
```

```
[1] -1
```

```
log2(80/20)
```

```
[1] 2
```

This log2 transformations has this property where if there is no change the log 2 value would be zero and if it doubles the log2 value will be 1, and if halved it will be -1.

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

```
head(meancounts)
```

```
  control.mean treated.mean      log2fc
1       900.75       658.00 -0.45303916
2         0.00         0.00         NaN
3       520.50       546.00  0.06900279
4       339.75       316.50 -0.10226805
5        97.25        78.75 -0.30441833
6         0.75         0.00        -Inf
```

We need to get rid of zero count genes that we cannot say anything about

> 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?

It makes the values all true. The unique function is to remove those outputs.

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

```
head(mycounts)
```

```
  control.mean      log2fc
1       900.75 -0.45303916
2         0.00         NaN
3       520.50  0.06900279
4       339.75 -0.10226805
5        97.25 -0.30441833
6         0.75        -Inf
```

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

There are 250 up-regulated genes.

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

Q9. Using the down.ind vector above can you determine how many down regulated genes we have at the greater than 2 fc level?

There are 367 genes down regulated.

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

No not fully because we don't know if the changes have a significant impact.

Setting up for DESeq

```
rownames(metadata) <- metadata$sample_id
```

```
rownames(metadata) <- metadata$id
```

```
metadata <- metadata[colnames(counts),]
```

```
counts <- counts[, -1]
```

```
metadata <- metadata[!is.na(metadata$id), ]
rownames(metadata) <- metadata$id
counts <- counts[, colnames(counts) != "ensgene"]
metadata <- metadata[colnames(counts), ]
```

```
all(colnames(counts) == rownames(metadata))
```

```
[1] TRUE
```

#7 Deseq analysis

```
#load up DESeq2
library(DESeq2)
```

```
Loading required package: S4Vectors
```

```
Loading required package: stats4
```

```
Loading required package: BiocGenerics

Loading required package: generics


Attaching package: 'generics'

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

    explain

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

    as.difftime, as.factor, as.ordered, intersect, is.element, setdiff,
    setequal, union


Attaching package: 'BiocGenerics'

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

    combine

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, is.unsorted, lapply, Map, mapply, match, mget,
    order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
    rbind, Reduce, rownames, sapply, saveRDS, table, tapply, unique,
    unsplit, which.max, which.min


Attaching package: 'S4Vectors'
```

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

    first, rename

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


Attaching package: 'IRanges'

The following objects are masked from 'package:dplyr':

    collapse, desc, slice

Loading required package: GenomicRanges

Loading required package: GenomeInfoDb

Loading required package: SummarizedExperiment

Loading required package: MatrixGenerics

Loading required package: matrixStats


Attaching package: 'matrixStats'

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

    count
```

```
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
```

```
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: NULL
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      padj
      <numeric>      <numeric> <numeric> <numeric> <numeric> <numeric>
1      747.1942     -0.3507030  0.168246 -2.084470 0.0371175  0.163035
2        0.0000             NA        NA        NA        NA        NA
3      520.1342      0.2061078  0.101059  2.039475 0.0414026  0.176032
4      322.6648      0.0245269  0.145145  0.168982 0.8658106  0.961694
5       87.6826     -0.1471420  0.257007 -0.572521 0.5669691  0.815849
...         ...             ...       ...       ...       ...       ...
38690  0.000000             NA        NA        NA        NA        NA
38691  0.000000             NA        NA        NA        NA        NA
38692  0.000000             NA        NA        NA        NA        NA
38693  0.974916      -0.668258   1.69456 -0.394354  0.693319        NA
38694  0.000000             NA        NA        NA        NA        NA
```

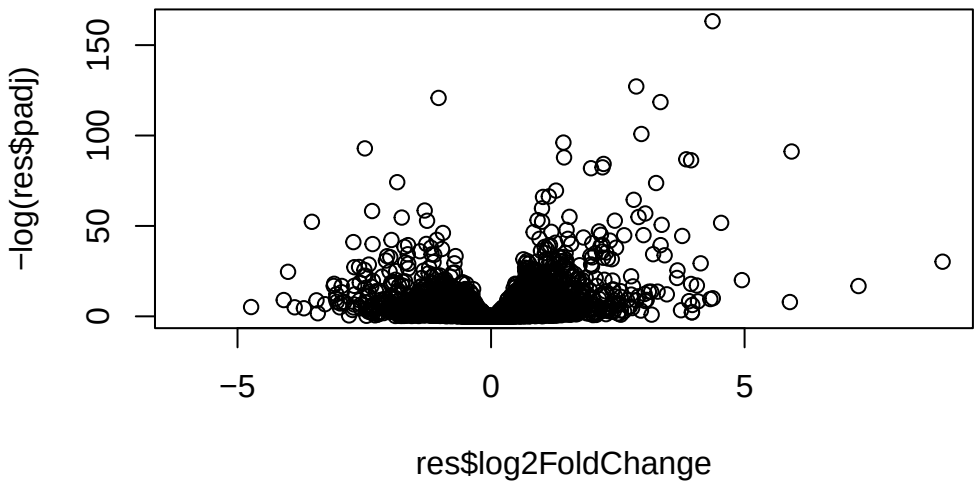
```
summary(res, alpha=0.05)
```

```
out of 25258 with nonzero total read count
adjusted p-value < 0.05
LFC > 0 (up)       : 1242, 4.9%
LFC < 0 (down)     : 939, 3.7%
outliers [1]       : 142, 0.56%
low counts [2]     : 9971, 39%
(mean count < 10)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

9. Volcano plot

Make a summary plot of our results

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




```
log(0.1)
```

```
[1] -2.302585
```

```
log(0.005)
```

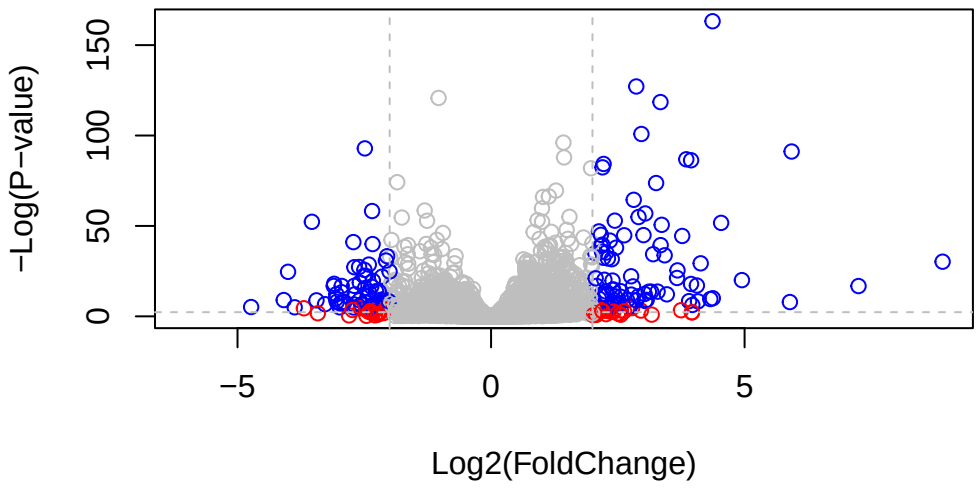
```
[1] -5.298317
```

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

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="gray", lty=2)
abline(h=-log(0.1), col="gray", lty=2)
```

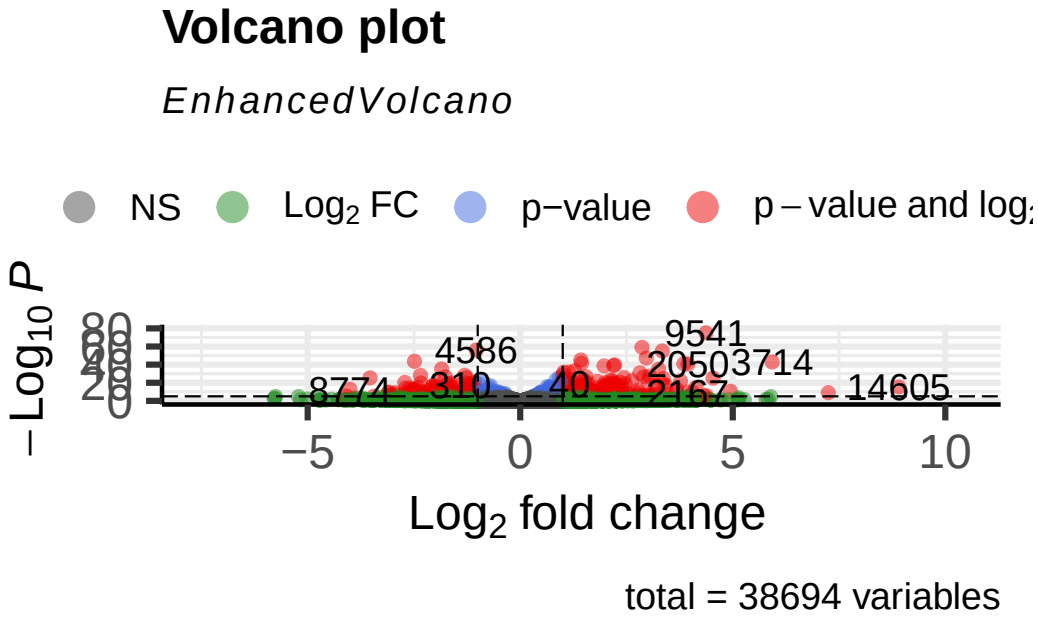



```
library(EnhancedVolcano)
```

```
Loading required package: ggrepel
```

```
x <- as.data.frame(res)

EnhancedVolcano(x,
    lab = rownames(x),
    x = 'log2FoldChange',
    y = 'pvalue')
```




```
summary(res)
```

```
out of 25258 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)       : 1563, 6.2%
LFC < 0 (down)     : 1188, 4.7%
outliers [1]       : 142, 0.56%
low counts [2]     : 9971, 39%
(mean count < 10)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

Saving our results

```
write.csv(res, file="DESeq2_results.csv")
```