Class_13_RNA_seq_analysis_DESeq2

Johann Tailor

Section 1 and 2

We will first set up the environment for performing RNA Seq Analysis:

install.packages("BiocManager") BiocManager::install()

For this class we will need DESeq2:

BiocManager::install("DESeq2")

```
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

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
```

```r
library("BiocManager")
```

Section 3

Reading the data into R:

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

Let's look at the data:

```
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
```

```
dim(counts)
```

```
[1] 38694     8
```

```
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
```

Questions: Q1. How many genes are in this dataset? Answer: 38694 genes are present in count dataset.

```
nrow(counts)
```

```
[1] 38694
```

Q2. How many samples? Answer: 8 samples are present

```
ncol(counts)
```

```
[1] 8
```

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

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

```
control treated
      4       4
```

```
# == looks for that particular pattern
sum(metadata$dex == "control")
```

```
[1] 4
```

Back check to see if all your controls are the same samples:

```
colnames(counts)
```

```
[1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"
[6] "SRR1039517" "SRR1039520" "SRR1039521"
```

```
metadata$id
```

```
[1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"
[6] "SRR1039517" "SRR1039520" "SRR1039521"
```

```
#Using == to check if they are perfectly same
colnames(counts) == metadata$id
```

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

Section 4:

Our plan: - First extract the controls (that is the columns) - Calculate the row wise mean (mean of counts for each gene)

Answer 3 and 4:

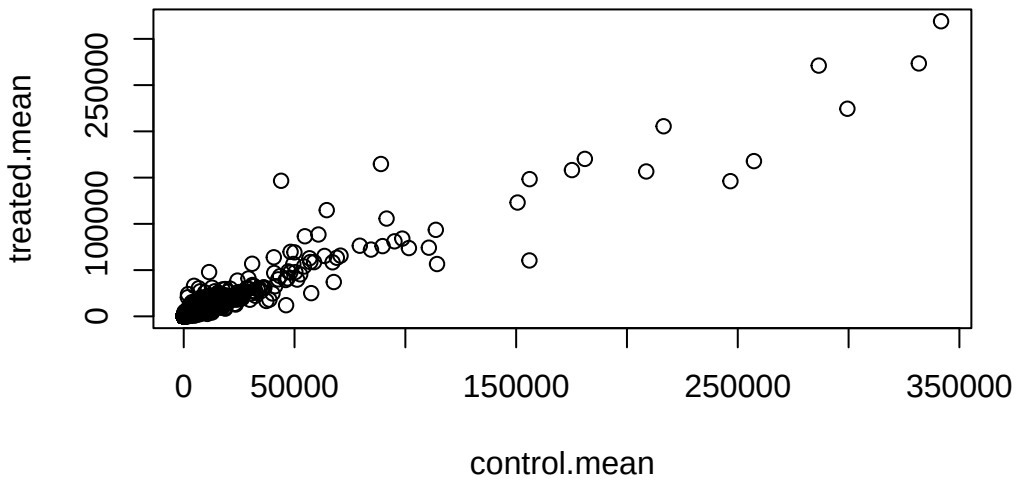
```
#Where are the control samples

#accessing the columns
control.inds <- metadata$dex == "control"
control.counts <- counts[,control.inds]
control.mean <- apply(control.counts, 1, mean)
```

```
#Where are the treated samples

#accessing the columns
treated.inds <- metadata$dex == "treated"
treated.counts <- counts[,treated.inds]

#getting means for each treated sample for each gene:
treated.mean <- apply(treated.counts, 1, mean)
```

Store these together:

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

```
plot(meancounts)
```



Answer 5: If we were to do this in ggplot, we would use geom_point()

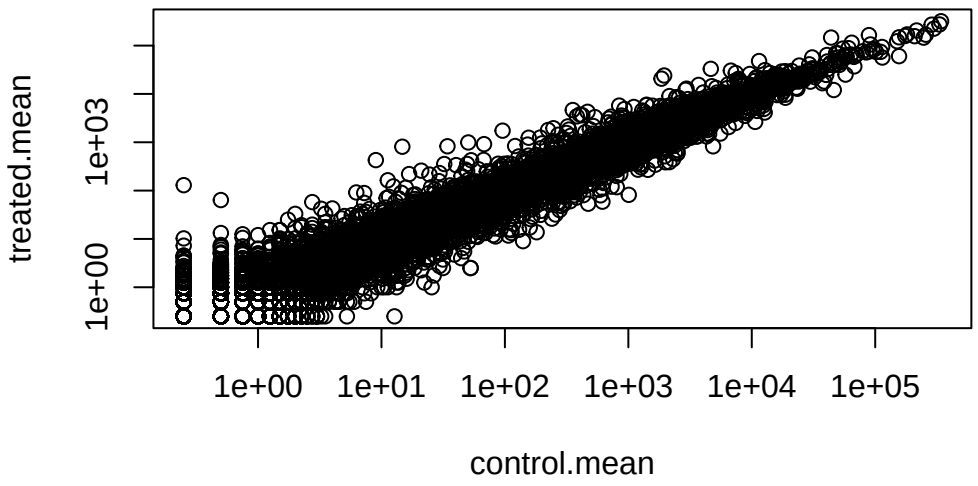
As it looks, our data is skewed the left and need to transformed for better resolution. To do this, we can use log scaling in the plot.

Answer 6: log = "xy"

```
plot(meancounts, log = "xy")
```

```
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
```



Lets determine the fold change now:

Fold change = Treated / Control

We will use log2 fold changes for our treated/control:

```
#the $log2fc adds another column to the meancounts dataset
meancounts$log2fc <- log2(meancounts$treated.mean/meancounts$control.mean)

head(meancounts)
```

```
                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
```

There are some 0.00 in treated.mean and control.mean which means that those genes are not affected. We want to remove these from the data set. How can we identify zero count genes in our 'meancounts'

```
#meancounts[,1:2] == "0"

#head(rowSums(meancounts[,1:2] == 0 )

#Only taking 1 and 2 column that has the means:

zero.sums <- rowSums(meancounts[,1:2] == 0)
to.rm.ind <- zero.sums > 0
mycounts <- meancounts[!to.rm.ind,]
```

Let's see how many genes are present:

```
nrow(mycounts)
```

```
[1] 21817
```

Answers: 8, 9 and 10

A common threshold for calling something "DE" is a log2 fold change of +2 and -2. So, lets how check how many of our genes are "up-regulated".

Upregulated genes:

```
sum(mycounts$log2fc >= +2)
```

```
[1] 314
```

Downregulated genes:

```
sum(mycounts$log2fc <= -2)
```

```
[1] 485
```

Answer 10: I don't trust these results because I don't know if they are significant.

Lets do this correct way;

```
library(DESeq2)
```

DESeq2 wants our data in a particular object called a deseq object and we can set this up within the DESeqpackage.

```
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
```

The main analysis function is called 'DESeq()' and we can now input our data:

```
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 our of the 'dds' we use the 'DESeq' function called 'results()'.

```
res <- results(dds)
head(res)
```

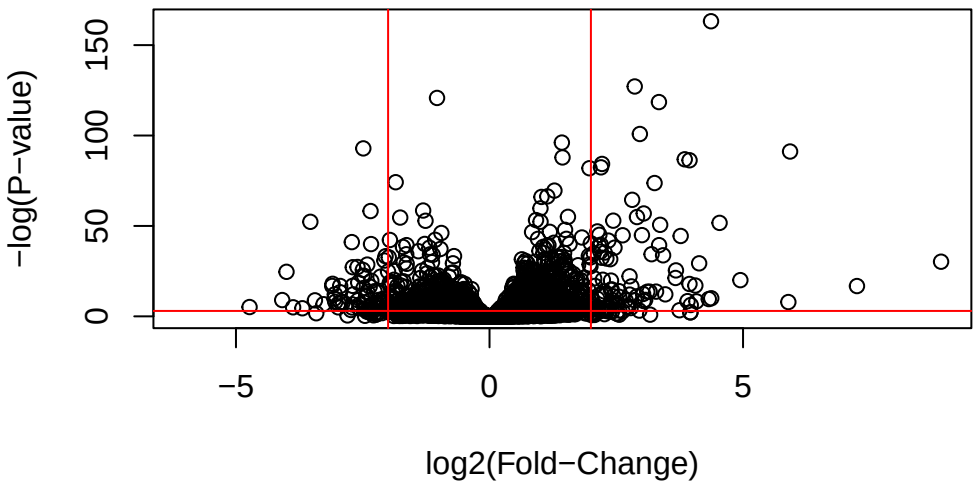
```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 6 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
                 <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195     -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 plot

A common visualization for this type of data is called a Volcano plot:

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



Saving our DESeq2 results:

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

Adding annotation to Ensemble IDs

You may have to install these with the BiocManager::install("AnnotationDbi") and BiocManager::install("org.Hs.eg.db") function calls.

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

Answer 11:

```
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
```

Now, lets also annotate with the EntrezID and save it as a new column

```
res$entrez <- mapIds(org.Hs.eg.db,
                     keys=row.names(res), # Our genenames
                     keytype="ENSEMBL",        # The format of our genenames
                     column="ENTREZID",          # 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 8 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
                <numeric> <character> <character>
```

```
ENSG00000000003  0.163035      TSPAN6        7105
ENSG00000000005        NA        TNMD       64102
ENSG00000000419  0.176032        DPM1        8813
ENSG00000000457  0.961694       SCYL3       57147
ENSG00000000460  0.815849       FIRRM       55732
ENSG00000000938        NA         FGR        2268
```

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