Class12- RNAseq Analysis

Loretta Cheng

Here we will use the DESeq2 package for RNASeq analysis. The data for today's class came from a study where airway smooth muscle cells were treated with dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al. 2014).

Import their Data

We need two things for this analysis

- **countData** (counts for every transcript/gene in each experiment)
- **colData** (metadata that describes the experimental setup)

```
countData <- read.csv("airway_scaledcounts.csv", row.names=1)
head(countData)
```

	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 <- read.csv("airway_metadata.csv")
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(countData)
```

```
[1] 38694
```

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

```
sum(metadata$dex == "control")
```

```
[1] 4
```

Step 1:

Calculate the mean of the control sample (i.e columns in countData) calculate the mean of the treated sample

(a) We need to find which columns in countData are "control" samples.

- look in the metadata (aka colData), $dex column

```
control.inds <- metadata$dex == "control"
```

(b) extract all the control columns from `countData` and call it `control.counts`

```
control.counts <- countData[ ,control.inds]
```

(c) calculate the mean value across the rows of `control.counts` i.e. calculate the mean count values for each gene in the control sample.

```
control.means <- rowMeans(control.counts)
head(control.means)
```

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

- Step 2. calculate the mean of the treated samples...

```
treated <- metadata[metadata[,"dex"]=="treated",]
treated.mean <- rowSums( countData[ ,treated$id] )/4
names(treated.mean) <- countData$ensgene
```

We now have control and treated mean count values. For ease of book-keeping I will combine these vectors into a new data.frame called `meancounts`

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

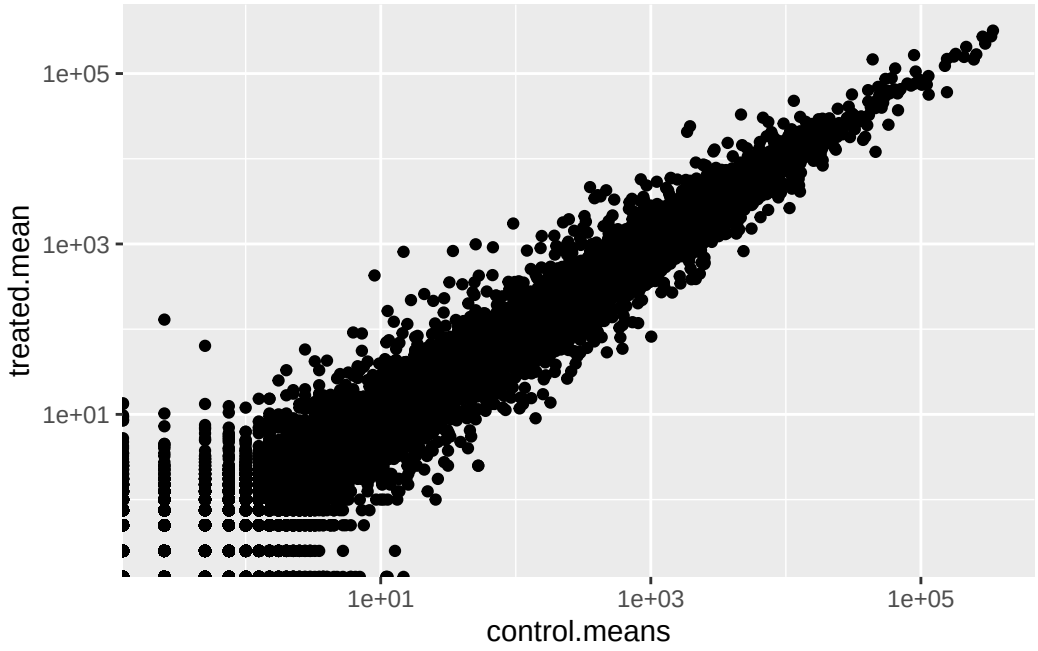
```
                control.means treated.mean
ENSG00000000003        900.75       658.00
ENSG00000000005          0.00         0.00
ENSG00000000419        520.50       546.00
ENSG00000000457        339.75       316.50
ENSG00000000460         97.25        78.75
ENSG00000000938          0.75         0.00
```

```
library(ggplot2)

ggplot(meancounts)+
  aes(control.means, treated.mean)+
  geom_point()+
  scale_x_log10()+
  scale_y_log10()
```

```
Warning: Transformation introduced infinite values in continuous x-axis
```

```
Warning: Transformation introduced infinite values in continuous y-axis
```



We will be using log10.

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

```
[1] 0
```

```
log2(20/10)
```

```
[1] 1
```

If I have half the amount I will have a log2 fold-change of -1.

```
log2(10/20)
```

```
[1] -1
```

```
log2(40/10)
```

```
[1] 2
```

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

```
                control.means 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
```

Q. How many genes are up regulated at the common threshold of +2 log2FC values?

```
sum(meancounts$log2fc >= 2, na.rm=TRUE)
```

```
[1] 1910
```

Hold on, what about stats! Yes these are big changes but are these changes significant!

To do this properly we will turn to the DESeq2 package.

DESeq2 analysis

```
#i message: false
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
```

To use DESeq we need our input contData and colData in a specific format that DESeq wants:

```
dds <- DESeqDataSetFromMatrix(countData= countData,
                        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
```

To run analysis I can now use the main DESeq2 function called `DESeq()` with `dds` as input.

```
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 out of this `dds` object we use the `results()` function from package

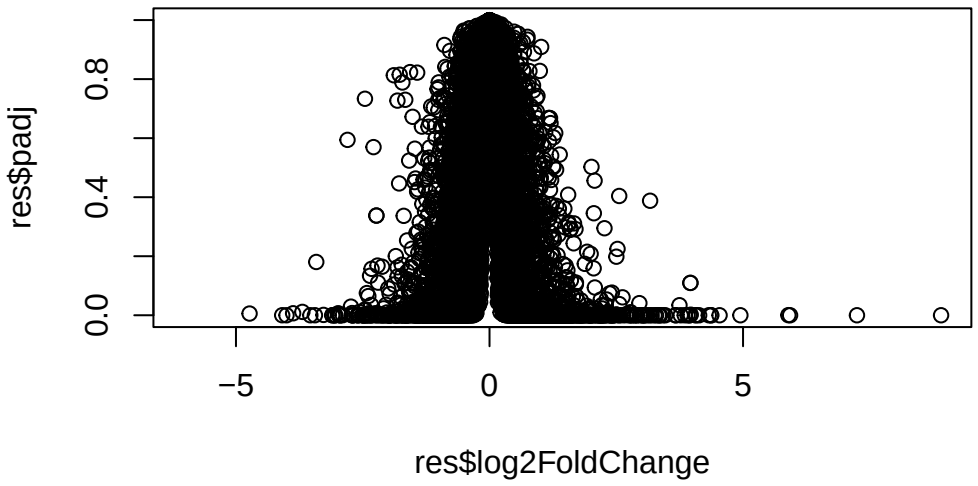
```
res <- results(dds)
head(res)
```

```
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

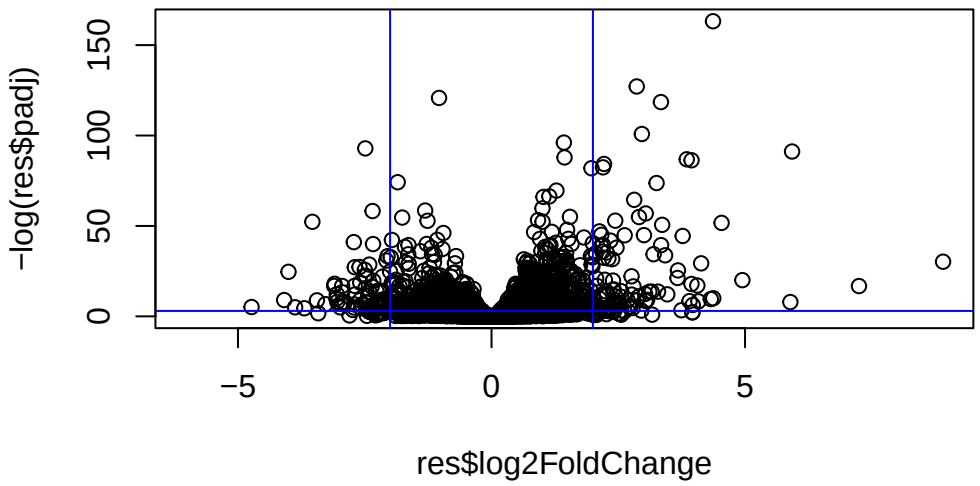
Let's make a final (for today) plot of the log2 fold-change vs the adjusted P-value.

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



It is the low P-values that we care about and these are lost in the skewed plot above. Let's take the log of the $padj values for our plot

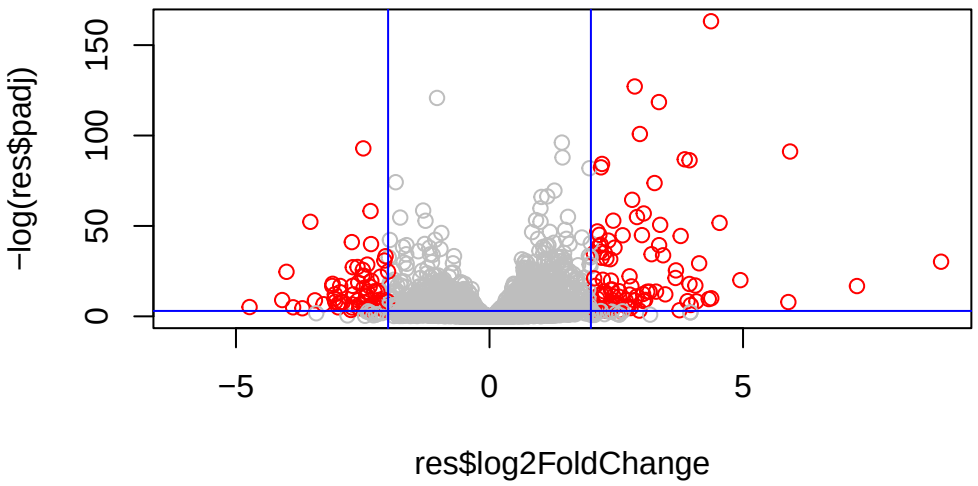
```
plot(res$log2FoldChange, -log(res$padj))
abline(v=c(+2, -2), col="blue")
abline(h= -log(0.05),col="blue")
```



Finally we can make a color vector to use in the plot to better highlight the genes we are about.

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

plot(res$log2FoldChange, -log(res$padj), col=mycols)
abline(v=c(+2, -2), col="blue")
abline(h= -log(0.05),col="blue")
```


−log(res$padj)
0
50
100
150
−5
0
5
res$log2FoldChange