Differential expression analysis using DESeq2

In this session, we will count reads aligning to genes and perform a differential expression analysis. First install and load all the packages required

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("DESeq2", version = "3.8")

install.packages("RColorBrewer")
install.packages("gplots")
```

```
library(DESeq2)
```

```
## Loading required package: S4Vectors

## Loading required package: stats4

## Loading required package: BiocGenerics

## Loading required package: parallel

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind,
##     colMeans, colnames, colSums, dirname, do.call, duplicated,
##     eval, evalq, Filter, Find, get, grep, grepl, intersect,
##     is.unsorted, lapply, lengths, Map, mapply, match, mget, order,
##     paste, pmax, pmax.int, pmin, pmin.int, Position, rank, rbind,
##     Reduce, rowMeans, rownames, rowSums, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which, which.max,
##     which.min

##
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:base':
##
##     expand.grid
```

```
## Loading required package: IRanges

## Loading required package: GenomicRanges

## Loading required package: GenomeInfoDb

## Loading required package: SummarizedExperiment

## 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")'.

## Loading required package: DelayedArray

## Loading required package: matrixStats

##
## Attaching package: 'matrixStats'

## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians

## Loading required package: BiocParallel

##
## Attaching package: 'DelayedArray'

## The following objects are masked from 'package:matrixStats':
##
##     colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges

## The following objects are masked from 'package:base':
##
##     aperm, apply
```

```
library(RColorBrewer)
library(gplots)
```

```
##
## Attaching package: 'gplots'

## The following object is masked from 'package:IRanges':
##
##     space
```

```
## The following object is masked from 'package:S4Vectors':
##
##     space

## The following object is masked from 'package:stats':
##
##     lowess
```

Set your working directory using setwd to where you have the counts file for day2 saved. Then import the counts file

```
counts <- read.delim("Day2_counts.txt",skip = 1)
```

The skip option tells R to skip the first line,which is a header line in the counts file. Now we need to clean up the data frame a bit. We need to remove the annotation (first 6 rows of the count dataframe), and clean up the sample names. The last line is removing the strings "Alignments.", and "_sorted.bam" from the column names

```
annotation <- counts[,c(1:6)]
rownames(counts) <- counts$Geneid
counts <- counts[,-c(1:6)]
colnames(counts) <- gsub("Alignments\\.|_sorted.bam","",colnames(counts))
```

Add some sample information.Here's its cell type, but this where you add all of your treatment groups.

```
cellType <- c(rep("Astrocytes",3),rep("Microglia",3))
sampleName <- paste(cellType,c(1:3),sep = "_")
colData <- data.frame(row.names = colnames(counts),
                      cellType = cellType,sampleName = sampleName)
colData
```

```
##              cellType   sampleName
## 15.007.001 Astrocytes Astrocytes_1
## 15.007.003 Astrocytes Astrocytes_2
## 15.007.004 Astrocytes Astrocytes_3
## 15.008.001  Microglia  Microglia_1
## 15.008.003  Microglia  Microglia_2
## 15.008.004  Microglia  Microglia_3
```

Combine the sample information with the read counts in a DESeq2DataObject

```
dds <- DESeqDataSetFromMatrix(countData = counts, design = ~ cellType, colData = colData)
```

The design option is used to specify which variables to include in your analysis..

Add in the annotation that we dropped from the count data frame

```
mcols(dds) <- DataFrame(annotation)
```

The function *mcols()* stands for metada columns, and the function *DataFrame()* is similar to the *data.frame()* function, but is specific to the certain object types found in biocondutor packages.

```
dds <- DESeq(dds,betaPrior = T)
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

The *DESeq()* function normalizes the read counts,estimates dispersions, and fits the linear model, all in one go. The betaPrior = T opton tells DESeq2 to squeeze the log Fold Changes of lowly expressed genes toward zero.

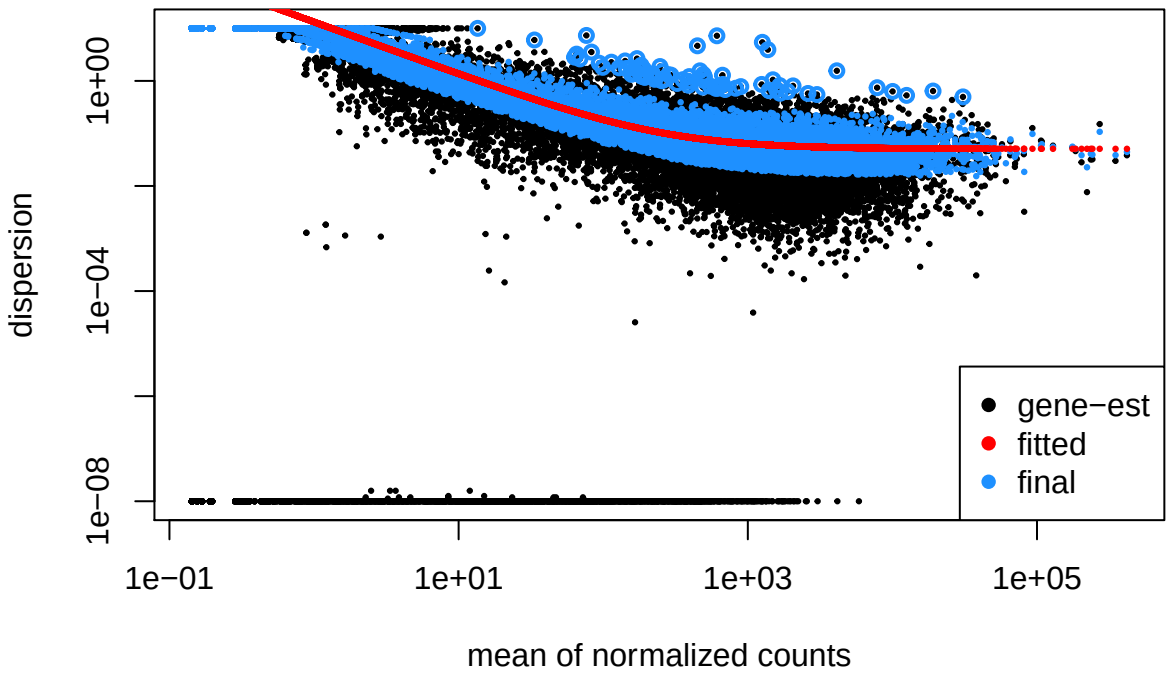
To see the normalization factors:

```
sizeFactors(dds)
```

```
## 15.007.001 15.007.003 15.007.004 15.008.001 15.008.003 15.008.004
##  1.1667199  0.8734766  0.8449063  0.9919398  1.0692580  1.1546788
```

Next step some exploratory data analysis. Plot the dispersions:

```
plotDispEsts(dds)
```




```
dev.copy2pdf(file = "dispEsts.pdf")
```

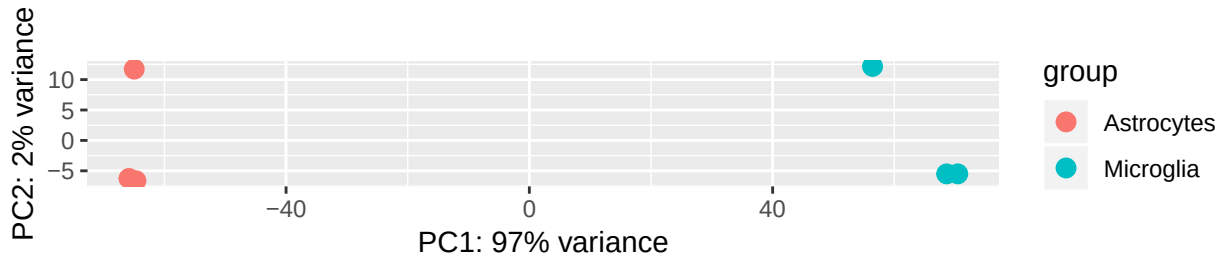
Each black dot in the plot represents the dispersion for one gene. The red line is fit to data (black bots), then the dispersions are squeezed toward the red line, resulting the final (blue) dispersion estimates. Usually, the dispersion is higest at the low counts and levels off at higher counts.

For the next two plots, we need to use the log of the normalized counts

```
rld <- rlog(dds)
```

One way of assessing how well your replicates group together is by creating a PCA (Principal Components Analyis) plot:

```
plotPCA(rld,intgroup = "cellType")
```




```
dev.copy2pdf(file = "PCA")
```

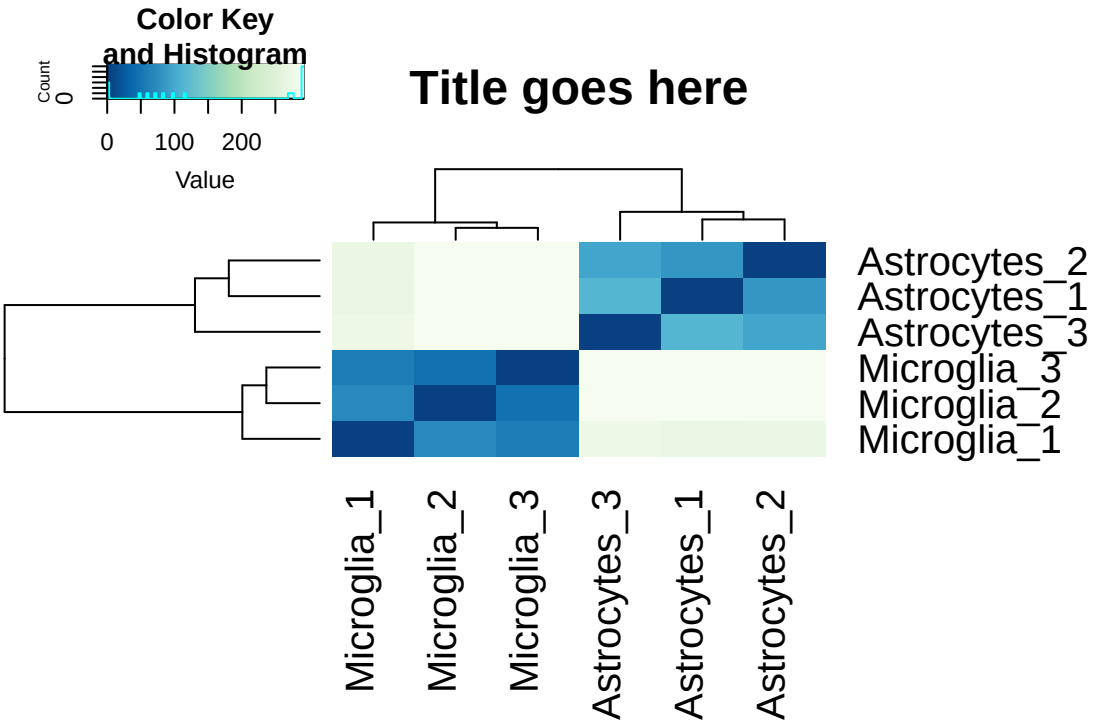
```
## pdf
##   2
```

Another way is generate a heatmap of samples

```
hmcol <- colorRampPalette(brewer.pal(9, 'GnBu'))(100)

dists <- dist(t(assay(rld)))
mat <- as.matrix(dists)
rownames(mat) <- colnames(mat) <- as.character(colData$sampleName)
hc <- hclust(dists)

heatmap.2(mat, Rowv=as.dendrogram(hc), symm=T, trace='none',
          col=rev(hmcol), margin=c(13,13), main='Title goes here')
```




```
dev.copy2pdf(file = "distClustering.pdf")
```

Now test for differential expression between treatment conditions

```
res <- results(dds,contrast = c("cellType","Astrocytes","Microglia"),alpha = 0.05)
```

In the contrast option, your reference or control, will be the last group ex “Microglia” -and all fold changes will be calculated relative to the Microglia, the alpha option is the FDR (False Discovery Rate)

Look at the results ordered by p-value

```
res[order(res$padj),]
```

```
## log2 fold change (MAP): cellType Astrocytes vs Microglia
## Wald test p-value: cellType Astrocytes vs Microglia
## DataFrame with 24062 rows and 6 columns
##                 baseMean   log2FoldChange             lfcSE
##                <numeric>        <numeric>         <numeric>
## Id4     5022.60840371119 8.14836272939039 0.252481775395744
## Scg3    4081.28084559166 7.37814424574032 0.232570157216413
## Fhl1    10321.2747029914 7.86139815689683 0.259112453150076
## Nrbp2   6411.07904932182 7.76854526769188 0.259027454728182
## Ddah1   11974.1172579159 7.34352524687506 0.256747426680861
## ...                  ...              ...               ...
## Gm21943                0               NA                NA
```

```
## Gm20816                0              NA                NA
## Gm20867                0              NA                NA
## Gm20806                0              NA                NA
## Gm20854                0              NA                NA
##                     stat               pvalue                 padj
##                <numeric>            <numeric>            <numeric>
## Id4     32.2730728450341 1.66996619271102e-228 2.89856032068851e-224
## Scg3    31.7243808666077 7.16542136082402e-221 6.21851092799112e-217
## Fhl1    30.3397156768207  3.4338521990606e-202 1.98671242063649e-198
## Nrbp2   29.9912041209841  1.2780005949578e-197 5.54556408167063e-194
## Ddah1   28.6021376798573 6.31978596099801e-180 2.19385049850085e-176
## ...                  ...                  ...                  ...
## Gm21943               NA                   NA                   NA
## Gm20816               NA                   NA                   NA
## Gm20867               NA                   NA                   NA
## Gm20806               NA                   NA                   NA
## Gm20854               NA                   NA                   NA
```

Get a summary of the results

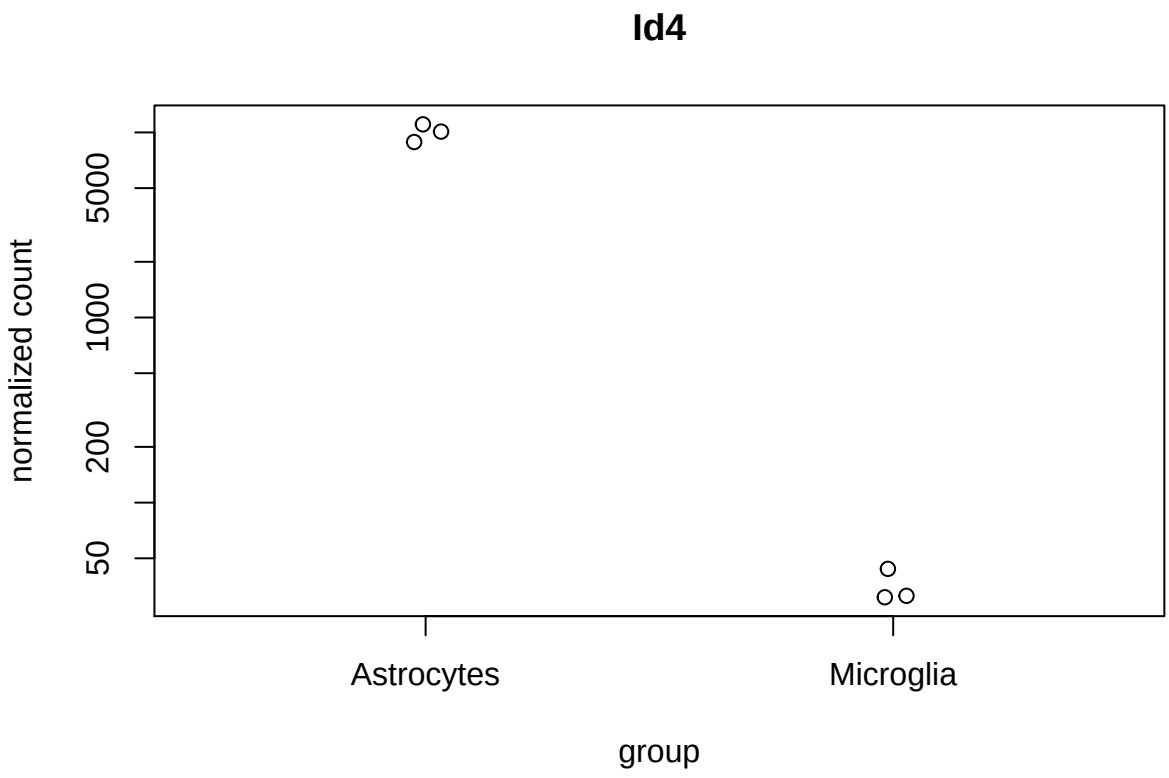
```
summary(res)
```

```
##
## out of 19183 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 6060, 32%
## LFC < 0 (down)     : 4930, 26%
## outliers [1]       : 0, 0%
## low counts [2]     : 1826, 9.5%
## (mean count < 1)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

The summary gives the number of genes that are up and down regulated, and the outlier genes, and genes removed due to low counts.

Plot the counts of some of the top differentially expressed genes as a sanity check:

```
plotCounts(dds,gene = "Id4",intgroup = "cellType")
```




```
dev.copy2pdf(file = "Id4_plotCounts.pdf")
```

This plot is a basic plot of what we did yesterday.

Also, you can subset the results table to look at your favourite genes:

```
subset(res,grepl("Hox",rownames(res)))
```

```
## log2 fold change (MAP): cellType Astrocytes vs Microglia
## Wald test p-value: cellType Astrocytes vs Microglia
## DataFrame with 41 rows and 6 columns
##                   baseMean    log2FoldChange             lfcSE
##                  <numeric>         <numeric>         <numeric>
## Hoxb13    1.38012381533628 -1.77248302329452  2.1680943606349
## Hoxb9                    0                NA                NA
## Hoxb8                    0                NA                NA
## Hoxb7    0.601041811804874 -1.56743314275305 2.30005646823577
## Hoxb6    0.197260527065998 0.639381112175818 2.30639168521003
## ...                    ...               ...               ...
## Hoxa9                    0                NA                NA
## Hoxa10    0.19080840071908 0.639381112175818 2.30639168521003
## Hoxa11                   0                NA                NA
## Hoxa11os                 0                NA                NA
## Hoxa13                   0                NA                NA
##                       stat            pvalue              padj
```

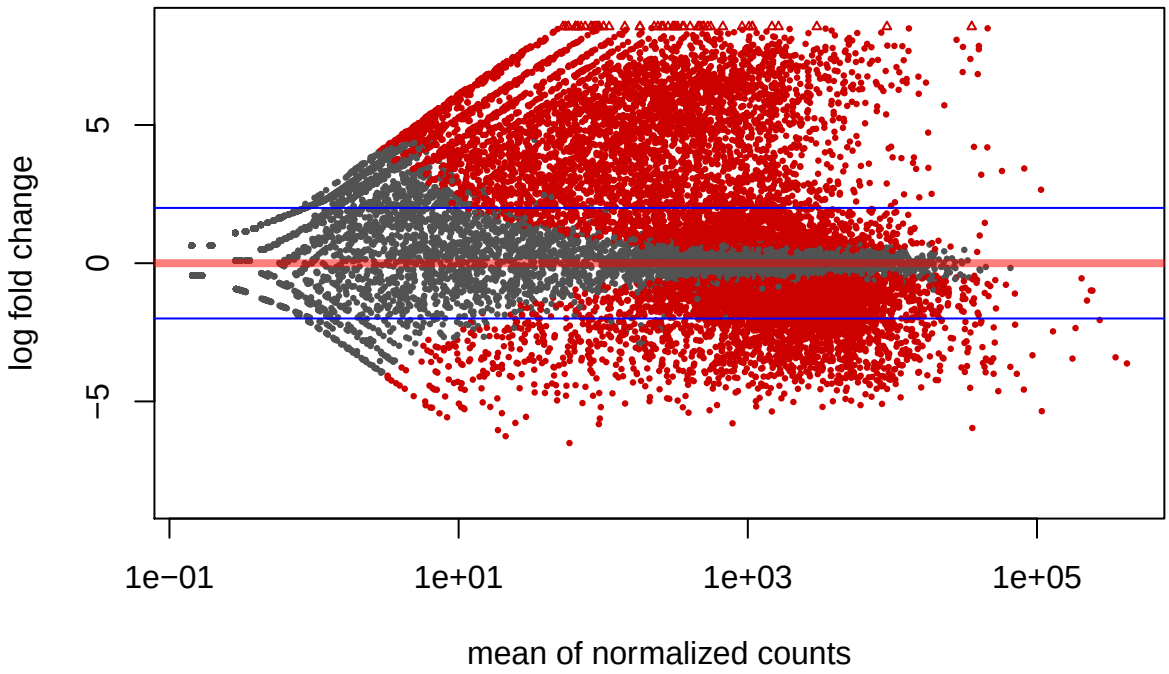
```
##                  <numeric>         <numeric>         <numeric>
## Hoxb13   -0.817530387734355 0.413625390789844 0.482804028778703
## Hoxb9                   NA                NA                NA
## Hoxb8                   NA                NA                NA
## Hoxb7    -0.681475939569139 0.495570386062094                NA
## Hoxb6     0.277221391438372  0.78161011508897                NA
## ...                     ...               ...               ...
## Hoxa9                   NA                NA                NA
## Hoxa10    0.277221391438372  0.78161011508897                NA
## Hoxa11                  NA                NA                NA
## Hoxa11os                NA                NA                NA
## Hoxa13                  NA                NA                NA
```

Save your results to a file:

```
write.csv(as.data.frame(res),file = "DEtable.csv")
```

Plot the log2 Fold Changes as function of expresson:

```
plotMA(res,alpha = 0.05)
abline(h=c(-2,2),col = "blue")
```




```
dev.copy2pdf(file = "MA.plot")
```

Finally record the version numbers of all the packages used by saving the results of sessionInfo() to a file:

```
capture.output(sessionInfo(),file = "sessionInfo.txt")
```

You can also save your DESeqDataObject and rlog objects:

```
save(dds,file = "DESeqData.rda")
save(rld, file = "rld.rda")
```

To load these objects in a future session you can use the load function:
load(file = “DESeqData.rda”)
load(file = “rld.rda”)