Lab practical:

1. Statistical Workflow in Differential Gene Expression Analysis

2. Experimental design

1. Statistical analysis workflow for detecting differentially expressed genes, using edgeR

The aim here is to get you familiar with the workflow. edgeR is an R package and part of the Bioconductor suite:

http://www.bioconductor.org/packages/release/bioc/html/edgeR.html

We will largely follow the edgeR user manual:

Installation:

```
source("http://bioconductor.org/biocLite.R")
biocLite("edgeR")
library(edgeR)
```

This might not work for newer versions of R. In that case try (already includes the next step, installing "baySeq"):

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

BiocManager::install(c("edgeR", "baySeq"))


library(baySeq)
library(edgeR)
```

Putting the data into correct format:

We'll work through an example dataset that is built into the package baySeq. This data set is a matrix (mobData) of counts acquired for three thousand small RNA loci from a set of Arabidopsis grafting experiments (generations of plants with different roots and shoots to look at how different signals are transported through the plant). baySeq is also a bioconductor package, and is also installed using:

```
source("http://bioconductor.org/biocLite.R")
biocLite("baySeq")
library(baySeq)
```

Download the data from StudentPortalen and load it into R's namespace with "load([[insert download directory here]]/"mobData.RData")". Or set your working directory with the data file in there and load it with this command:

```
load("mobData.RData")
head(mobData)  # Look at first lines of data

help(mobData)  # Check info about this dataset. The Work was published in Science in 2010.

mobDataGroups <- c("MM", "MM", "WM", "WM", "WW", "WW")  # Make a vector that gives a name to the treatments. This is needed to identify the treatment groups in the expression dataset.
```

edgeR works on a table of integer read counts, with rows corresponding to genes and columns to independent libraries. edgeR stores data in a simple list-based data object called a DGEList. This type of object is easy to use because it can be manipulated like any list in R. You can make this in R by specifying the counts and the groups in the function DGEList().

```
d <- DGEList(counts=mobData,group=factor(mobDataGroups))  # Note that here we use the new vector that specifies which samples are which.

d  # Look at the new DGEList object we created. Shows how the expression values are stored under $counts and sample groups are specified under $samples
```

The philosophy of this package is that all the information should be contained in a single variable. When we use functions from this package later, we will write something like d <- estimateCommonDisp(d) which we normally would guess is overwriting d. Instead, this function passes everything that was already in d through the function but just adds one more element to the list.

Filtering the data:

Idea is to remove genes that are not expressed. We can choose this cutoff by saying we must have at least X counts per million (calculated with cpm() in R) on any particular gene that we want to keep. In this example, we're keeping a gene if it has a cpm of 100 or greater for at least two samples. This is a very high threshold and used here just for illustrative purposes.

```
d.full <- d # Give the object new name, to keep it for answering the question.
head(d$counts) # First rows of the count data

head(cpm(d)) # The same by showing counts adjusted to the library size by calculating counts per million reads

apply(d$counts, 2, sum) # Total counts per sample


keep <- rowSums(cpm(d)>100) >= 2 # Specify rows that meet our filtering criteria
d <- d[keep,] # Subset the original data to only keep the rows we want
```

After filtering, reset the library sizes:

```
d$samples$lib.size <- colSums(d$counts) # Re-compute the library size since now they have fewer tags.
d$samples
```

Question: How many genes did you filter out? What happens if you do the most typical filtering of keeping any gene that has at least 1 count per million reads in at least two samples? How about if you keep only very highly expressed genes, say those with 1000 copies or more? Tip! You can look at the numbers retained easily by looking at the dimensions of your filtered data with dim()

Normalizing the data:

EdgeR is concerned with differential expression analysis rather than with the quantification of expression levels. It is concerned with relative changes in expression levels between conditions, but not directly with estimating absolute expression levels. There are two things that edgeR normalizes: the differences in library sizes and differences among genes (tags) within libraries that are due to disproportionate effect of some highly expressed genes, in only some samples, leaving little sequencing depth on other lowly expressed genes, which can lead to false conclusions when comparing samples with and without such effects that bias the expression counts.

The calcNormFactors() function normalizes for RNA composition by finding a set of scaling factors for the library sizes that minimize the log-fold changes between the samples for most genes. The default method for computing these scale factors uses a trimmed mean of M-values (TMM) between each pair of samples. We call the product of the original library size and the scaling factor the effective library size. The effective library size replaces the original library size in all downsteam analyses.

```
d <- calcNormFactors(d)
```
#Without this, the default value is 1 for all values in d$samples$norm.factors.

```
d
```

Data exploration:

Before proceeding with the computations for differential expression, it is a good idea to produce a plot showing the sample relations based on multidimensional scaling (a PCA plot or a clustering based plot would be alternative possibilities). Here is an example.

```
plotMDS(d, method="bcv", col=as.numeric(d$samples$group))
legend("bottomleft", as.character(unique(d$samples$group)),
col=1:3, pch=20)
```

Question: What can we learn from this kind of plot?

Estimation of dispersion:

The first major step in the analysis of DGE data using the negative binomial model is to estimate the dispersion parameter for each tag, a measure of the degree of inter-library variation for that tag. Dispersion variation is the additional variation beyond Poisson, and mostly arises from biological differences among samples, so it represents the variation among replicates and is therefore vital to estimate as a measure of sampling error.

```
d1 <- estimateCommonDisp(d, verbose=T)
```
Estimates the common dispersion using all genes.

```
Disp = 0.07776, BCV = 0.2789
```

Estimating the common dispersion gives an idea of overall variability across the genome for this dataset. E.g. dispersion =0.19 —> sqrt(dispersion)= BVC = 0.44. This means that the *true* estimated expression values vary up and down by 44% between replicates.

In a routine differential expression analysis, we use empirical Bayes tagwise dispersions. This estimates the dispersion variances *per gene*. Note that common dispersion needs to be estimated before estimating tagwise dispersions.

```
d1 <- estimateTagwiseDisp(d1)
```
Estimates the dispersion per gene, using the replicates

```
plotBCV(d1)
```
Plots the tag wise biological coefficient of variation (square root of dispersions) against log2-CPM.

Question: Is common dispersion a fair representation of the dispersion for every gene in this dataset?

GLM estimates of dispersion

You can also estimate the dispersion directly using a GLM model. For this you first fit a trended model (if you do not fit a trend, the default is to use the common dispersion as a trend). This considers the gene to gene differences in dispersion and seeks to find a common trend. Then you can fit the tagwise dispersion which is a function of this model. Look at the different dispersion estimates in a plot. Now the taqwise dispersion takes information from the common and trended dispersion estimates to "shrink" the gene-wise estimates towards a common trend.

When we do this estimation with a GLM we have to first tell the model how the data is structured: how the replicates relate to each treatment.

```
design.mat <- model.matrix(~ 0 + d$samples$group)
colnames(design.mat) <- levels(d$samples$group)

d2 <- estimateGLMCommonDisp(d,design.mat)
d2 <- estimateGLMTrendedDisp(d2,design.mat, method="power")

d2 <- estimateGLMTagwiseDisp(d2,design.mat)
plotBCV(d2)
```

GLM testing for differential expression:

Now we get to strat asking the biological questsions of how many differentially expressed we can find. Here we fit the generalized linear model first and then apply likelihood ratio test on specific contrasts of interest.

```
fit <- glmFit(d2, design.mat)

# Contrasts:
lrt12 <- glmLRT(fit, contrast=c(1,-1,0))
lrt23 <- glmLRT(fit, contrast=c(0,1,-1))
lrt13 <- glmLRT(fit, contrast=c(1,0,-1))

topTags(lrt12, n=10) # Look at the top 10 most significant genes.

de2 <- decideTestsDGE(lrt12, adjust.method="BH", p.value =
0.05)
summary(de2)  # Get a summary of up and downregulated genes
```

Questions:

1. How many genes are affected by mutations in the shoot and in what way?

Tips:

- Which contrast is informative for this? Look at the design matrix to understand the contrasts.

```
Reminder!
MM="triple mutant shoot grafted onto triple mutant root"
WM="wild-type shoot grafted onto triple mutant root"
WW="wild-type shoot grafted onto wild-type root"
```

- How many genes are up- vs. Down-regulated in each group involved in the contrast?

2. How does multiple testing method matter to these results? What happens if you don't adjust at all? What does the difference imply?

Tip:

- Check ?decideTestsDGE to see how to try out different p-value adjustment methods.

3. How many genes in this contrast have expression difference greater than 2-fold?

Tip:

- Fold difference is calculated here in log2 scale.
- Check again ?decideTestsDGE to see how you can introduce a cut-off value for the expression difference (doesn't matter if genes are up- or down-regulated in a given contrast)

2. Experimental design: How replication affects power

Here your task is to analyse the same dataset using the same statistical pipeline but using a different number of replicates (N=6, N=4, N=2), to test how the replication affects your ability to find differentially expressed genes.

Read in the full data and create a vector to identify the treatment groups:

```
x <- read.delim("HumanCellLineExps_CompRepsEdited.txt",
row.names="Gene")

Design <-
c("Treatment","Treatment","Treatment","Treatment","Treatment",
"Treatment", "Control", "Control", "Control", "Control",
"Control", "Control")
```

Using the pipeline created above, test how many differentially expressed genes do you find between the treatment and control samples when using six, four and two replicates for treatment and control groups. Create a barplot to illustrate your findings.

Tip: The pipeline needs to contain the following steps:

1. Create the DGEList object.
2. Filter the data to keep only expressed genes (>1cpm in >=2 samples)
3. Re-compute the library size.
4. Normalize the libraries (calcNormFactors).
5. Make a design matrix (model.matrix).
6. Estimate dipersion (estimateDisp, which calculates common, taqwise and trended simultanously). Note that this is speedier way to do simulatenously the three steps we tried out above.
7. Fit the GLM model (glmFit).
8. Test the contrast Control vs. Treatment (glmLRT).
9. Look at the top 10 genes (topTags).
10. Get a summary of the significant genes (summary, decideTestsDGE, adj.method="BH", p.value=0.05).

For the barplot:

a. Create a vector for the significant genes found with N=2, N=4, N=6 samples.
b. Name them (names())
c. Make a plot to visualize the differences (barplot())