edgeR: differential expression analysis of digital gene expression data

User's Guide

Mark Robinson, Davis McCarthy,
Yunshun Chen, Gordon K. Smyth

23 March 2012

Contents

1 Introduction 3

2 How to get help 3

3 Quick start 4

4 Negative binomial models 4

5 Reading data 5

6 Normalization issues for counts data 6
6.1 General comments 6
6.2 Calculating normalization factors 7

7 Pairwise comparisons between group (classic) 8
7.1 Estimating dispersions 8
7.2 Testing for DE genes 10

8 General experiments (glm functionality) 11
8.1 Estimating dispersions 11
8.2 Testing for DE genes 12

9 What to do if you have no replicates 13

10 Adjustments for gene length, GC content, mappability and so on 14

11 Case study: SAGE Profiles of Normal and Tumour Tissue 15
11.1 Introduction 15
11.2 Source of the data 15
11.3 Reading in the data and creating a `DGEList` object 15
11.4 Normalization and filtering 16
11.5 Estimating the dispersion 17
11.6 Differential expression 19
11.7 Setup 22

12 Case study: deepSAGE of wild-type vs Dclk1 transgenic mice 23
12.1 Introduction 23
12.2 Source of the data 23
12.3 Reading in the data and creating a `DGEList` object 24
12.4 Data exploration 25
12.5 Estimating the dispersion 26
12.6 Differential expression 27
12.7 Setup 29

13 Case Study: RNA-seq of Hormone-Treated LNCaP Cells 30
13.1 Introduction 30
13.2 Source of the data 30
13.3 Reading in the data and creating a `DGEList` object 31
13.4 Normalization and filtering 31
13.5 Data exploration 32
13.6 Estimating the dispersion 33
13.7 Differential expression 34
13.8 Setup 36

14 Case study: Oral carcinomas vs matched normal tissue 37
14.1 Introduction 37
14.2 Source of the data 37
14.3 Reading in the data and creating a `DGEList` object 37
14.4 Normalization and filtering 39
14.5 Data exploration 40
14.6 The design matrix 40
14.7 Estimating the dispersion 41
14.8 Differential expression 43
14.9 Setup 44

15 Case study: pathogen inoculated arabidopsis with batch effects 46
15.1 Introduction 46
15.2 RNA samples 46
15.3 Sequencing 46
15.4 Filtering and normalization 46
15.5 Data exploration 47
15.6 The design matrix 49
15.7 Estimating the dispersion 49
15.8 Differential expression 50
15.9 Setup 52

1 Introduction

This guide provides an overview of the Bioconductor package edgeR for differential expression analyses of RNA-Seq, SAGE or "next generation" sequencing data [Robinson et al., 2010]. The guide includes a number of fully worked case studies. The package can be applied to any technology that produces read counts for genomic features. Of particular interest are summaries of short reads from deep sequencing technologies such as Illumina^{TM}, 454 or ABI SOLiD applied to RNA-Seq, SAGE-Seq or ChIP-Seq experiments. edgeR provides statistical routines for assessing differential expression in RNA-Seq experiments or differential marking in ChIP-Seq experiments.

The package implements exact statistical methods for multigroup experiments developed by Robinson and Smyth [2007, 2008]. It also implements statistical methods based on generalized linear models, suitable for multifactor experiments of any complexity, developed by McCarthy et al. [2012]. Sometimes we refer to the former exact methods as *classic* edgeR, and the latter as *glm* edgeR. However the two sets of methods are complementary and can often be combined in the course of a data analysis. Most of the glm functions can be identified by the letters "`glm`" as part of the function name.

A particular feature of edgeR functionality, both classic and glm, are empirical Bayes methods that permit the estimation of gene-specific biological variation, even for experiments with minimal levels of biological replication.

edgeR can be applied to differential expression at the gene, exon, transcript or tag level. In fact, read counts can be summarized by any genomic feature. edgeR analyses at the exon level are easily extended to detect differential splicing or isoform-specific differential expression.

2 How to get help

Most questions about edgeR will hopefully be answered by the documentation or references. Every function mentioned in this guide has its own help page. For example, a detailed description of the arguments and output of the exactTest function can be read by typing ?exactTest or help(exactTest) at the R prompt.

The authors of the package always appreciate receiving reports of bugs in the package functions or in the documentation. The same goes for well-considered suggestions for improvements. Other questions about how to use edgeR are best sent to the Bioconductor mailing list bioconductor@stat.math.ethz.ch. Often other users are likely to have experienced similar problems, and posting to the list allows everyone to gain from the answers. To subscribe to the mailing list, see https://stat.ethz.ch/mailman/listinfo/bioconductor. Please send requests for general assistance and advice to the mailing list rather than to the individual authors. Users posting to the mailing list for the first time may find it helpful to read the posting guide at http://www.bioconductor.org/doc/postingGuide.html.

3 Quick start

A classic edgeR analysis might look like the following. Here we assume there are four RNA-Seq libraries in two groups, and the counts are stored in a text file, with gene symbols in a column called Symbol.

```
> x <- read.delim("fileofcounts.txt",row.names="Symbol")
> group <- factor(c(1,1,2,2))
> y <- DGEList(counts=x,group=group)
> y <- estimateCommonDisp(y)
> y <- estimateTagwiseDisp(y)
> et <- exactTest(y)
> topTags(et)
```

A glm edgeR analysis of the same data would look similar, except that a design matrix would be formed:

```
> design <- model.matrix(~group)
> y <- estimateGLMTrendedDisp(y,design)
> y <- estimateGLMTagwiseDisp(y,design)
> fit <- glmFit(y,design)
> lrt <- glmLRT(y,fit,coef=2)
> topTags(lrt)
```

Many variants are available on this analysis.

4 Negative binomial models

The starting point for an RNA-Seq experiment is a set of n RNA samples, typically associated with a variety of treatment conditions. Each sample is sequenced, short reads are mapped to the appropriate genome, and the number of reads mapped to each genomic feature of interest is recorded. The number of reads from sample i mapped to gene g will be denoted

y_{gi}. The set of genewise counts for sample i makes up the expression profile or *library* for that sample. The expected size of each count is the product of the library size and the relative abundance of that gene in that sample.

Two levels of variation can be distinguished in any RNA-Seq experiment. First, the relative abundance of each gene will vary between RNA samples, due mainly to biological causes. Second, there is measurement error, the uncertainty with which the abundance of each gene in each sample is estimated by the sequencing technology. If aliquots of the same RNA sample are sequenced, then Marioni et al. [2008] claimed that the read counts for a particular gene should vary according to a Poisson law. If sequencing variation is Poisson, then it can be shown (Methods) that the squared coefficient of variation (CV) of each count between biological replicate libraries is the sum of the squared CVs for technical and biological variation respectively,

$$\text{Total CV}^2 = \text{Technical CV}^2 + \text{Biological CV}^2.$$

Biological CV (BCV) is the coefficient of variation with which the (unknown) true abundance of the gene varies between replicate RNA samples. It represents the CV that would remain between biological replicates if sequencing depth could be increased indefinitely. The technical CV decreases as the size of the counts increases. BCV on the other hand does not. BCV is therefore likely to be the dominant source of uncertainty for high-count genes, so reliable estimation of BCV is crucial for realistic assessment of differential expression in RNA-Seq experiments. If the abundance of each gene varies between replicate RNA samples in such a way that the genewise standard deviations are proportional to the genewise means, a commonly occurring property of measurements on physical quantities, then it is reasonable to suppose that BCV is approximately constant across genes. We allow however for the possibility that BCV might vary between genes and might also show a systematic trend with respect to gene expression or expected count.

The magnitude of BCV is more important than the exact probabilistic law followed by the true gene abundances. For mathematical convenience, we assume that the true gene abundances follow a gamma distributional law between replicate RNA samples. This implies that the read counts follow a negative binomial probability law.

5 Reading data

edgeR requires three pieces of information:

1. `counts`: a matrix of counts where each row represents a gene (or whatever genomic feature is being tracked) and each column is a different sample.

2. `group` factor or `design` matrix: for classic edgeR, a factor of length `ncol(counts)` giving the experimental group, for glm edgeR, a design matrix indicating the assignment of treatments to samples.

3. lib.size: vector of length ncol(counts) giving the total number of reads sequenced for each sample. If not separately provided, will be set to colSums(counts).

We assume that the counts are stored in one of two formats. Either there is a single file containing a table of counts with the first column containing the tag identifiers and the remaining columns containing the tag counts for each library sequenced, or there is an individual file for each library, each with first column for tag identifiers and second column for counts.

If the counts for all libraries are stored in a single file, then an appropriate in-built R function (such as read.delim or read.csv) can be used to read the table of counts into R. See the help documentation (?DGEList or ?"DGEList-class") or the examples below for further details.

If the counts are stored in separate files, then, given a vector containing the filenames the edgeR function readDGE will read in the data from the individual files, collate the counts into a table and compute the library sizes and return a DGEList object. See the help documentation (?readDGE) or the examples below for further details.

Here is a simple example of creating a DGEList object given a count matrix:

```
> group <- factor(c(0,0,0,1,1,1))
> D <- DGEList(y, group=group)
```

6 Normalization issues for counts data

6.1 General comments

The edgeR methodology needs to work with the original digital expression counts, so these should not be transformed in any way by users prior to analysis. edgeR automatically takes into account the total size (total read number) of each library in all calculations of fold-changes, concentration and statistical significance. For some datasets, no other normalization is required for evaluating differential expression.

It bears emphasizing that RPKM values should *not* be used for assessing differential expression of genes between samples in edgeR. We use the raw counts, because the methods implemented in edgeR are based on the negative binomial distribution, a discrete distribution. To be able to perform good inference on differential expression it is very important to model the mean-variance relationship in the data appropriately. There are good reasons why the NB model is appropriate for the raw count data, but transforming the data using RPKM (or FPKM or similar) renders our distributional assumptions invalid and we cannot guarantee that our methods will be reliable for such transformed data.

There are methods implemented in edgeR to normalize the counts for compositional bias in sequenced libraries and for differences between libraries in sequencing depth. These adjustments are offsets in the models used for testing DE and do not transform the counts in any way.

The reason we do not worry about gene length bias, GC bias and so on when conducting DE analysis of the same genes *between* samples is that we expect (and hope) that the biases will affect the same gene in the same way in different samples. This being the case, then it is OK to test for DE gene between samples because such biases in effect "cancel out" when making the comparison between samples. This reasoning does not hold for comparing the expression level of *different genes* in *one sample*—to do this you would probably need to account for gene length and other biases, but this is not what `edgeR` is designed to do.

6.2 Calculating normalization factors

Recently, Robinson and Oshlack [2010] described a method to account for a bias introduced by what they call RNA composition. In brief, there are occasions when comparing different DGE libraries where a small number of genes are very highly expressed in one sample, but not in another. Because these highly expressed genes consume a substantial proportion of the sequencing "real estate", the remaining genes in the library are undersampled. Similarly, this situation may occur when the two tissues being compared have transcriptomes of different sizes, i.e. when there are noticeably more transcripts expressed in one tissues than the other. Robinson and Oshlack [2010] show that in comparing kidney and liver RNA, there are a large number of genes expressed in kidney but not in liver, causing the remaining genes to be undersampled and skewing the differential expression calls. To account for this, the authors developed an empirical approach to estimate the bias and proposed to build that into the library size (or, an offset in a generalized linear model), making it an *effective* library size. We demonstrate this below on the Marioni et al. [2008] RNA-seq dataset.

Given a table counts or a `DGEList` object, one can calculate normalization factors using the `calcNormFactors()` function.

```
> head(D)

   R1L1Kidney R1L2Liver R1L3Kidney R1L4Liver
10          0         0          0         0
15          4        35          7        32
17          0         2          0         0
18        110       177        131       135
19      12685      9246      13204      9312
22          0         1          0         0

> g <- gsub("R[1-2]L[1-8]", "", colnames(D))
> d <- DGEList(counts = D, group = substr(colnames(D), 5, 30))
> d$samples

            group lib.size norm.factors
R1L1Kidney Kidney  1804977            1
R1L2Liver   Liver  1691734            1
R1L3Kidney Kidney  1855190            1
R1L4Liver   Liver  1696308            1
```

```
> d <- calcNormFactors(d)
> d$samples

             group lib.size norm.factors
R1L1Kidney Kidney  1804977        1.209
R1L2Liver   Liver  1691734        0.821
R1L3Kidney Kidney  1855190        1.225
R1L4Liver   Liver  1696308        0.823
```

By default, `calcNormFactors` uses the TMM method and the sample whose 75%-ile (of library-scale-scaled counts) is closest to the mean of 75%-iles as the reference. Alternatively, the reference can be specified through the `refColumn` argument. Also, you can specify different levels of trimming on the log-ratios or log-concentrations, as well as a cutoff on the log-concentrations (See the help documentation for further details, including other specification of estimating the normalization factors).

To see the bias and normalization visually, consider a smear plot between the first (kidney) and second (liver) sample. In the left panel of Figure 1, we show a smear plot (X-axis: log-concentration, Y-axis: log fold-change of liver over kidney, those with 0 in either sample are shown in the smear on the left) of the raw data (Note: the argument `normalize=TRUE` *only* divides by the sum of counts in each sample and has nothing to do with the normalization factors mentioned above). One should notice a shift downward in the log-ratios, presumably caused by the genes highly expressed in liver that are taking away sequencing capacity from the remainder of the genes in the liver RNA sample. The red line signifies the estimated TMM (trimmed mean of M values) normalization factor, which in this case represents the adjustment applied to the library size to account for the compositional bias. The right panel of Figure 1 simply shows the M and A values after correction. Here, one should find that the bulk of the M-values are centred around 0.

```
> par(mfrow = c(1, 2))
> maPlot(d$counts[, 1], d$counts[, 2], normalize = TRUE, pch = 19,
+     cex = 0.4, ylim = c(-8, 8))
> grid(col = "blue")
> abline(h = log2(d$samples$norm.factors[2]/d$samples$norm.factors[1]),
+     col = "red", lwd = 4)
> eff.libsize <- d$samples$lib.size * d$samples$norm.factors
> maPlot(d$counts[, 1]/eff.libsize[1], d$counts[, 2]/eff.libsize[2],
+     normalize = FALSE, pch = 19, cex = 0.4, ylim = c(-8, 8))
> grid(col = "blue")
```

7 Pairwise comparisons between group (classic)

7.1 Estimating dispersions

When a negative binomial model is fitted, we need to estimate the dispersion(s) before we carry out the analysis. edgeR uses the quantile-adjusted conditional maximum likelihood

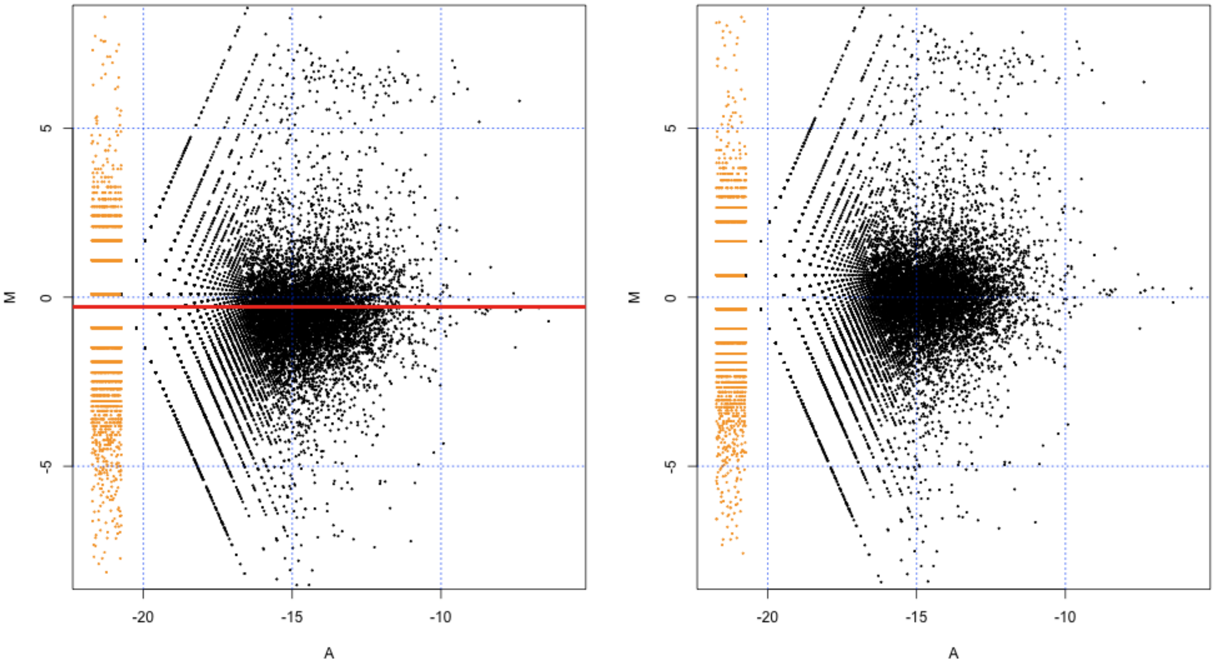


Figure 1: Smear plots before (left) and after (right) composition normalization.

(qCML) method to experiments with single factor.

Compared against several other estimators (e.g. maximum likelihood estimator, Quasi-likelihood estimator etc.) using an extensive simulation study, qCML is the most reliable in terms of bias on a wide range of conditions and specifically performs best in the situation of many small samples with a common dispersion, the model which is applicable to Next-Gen sequencing data. We have deliberately focused on very small samples due to the fact that DNA sequencing costs prevent large number of replicates for SAGE and RNA-seq experiments.

The qCML method can estimate a common dispersion for all the tags or separate dispersions for individual tags. As individual tags typically don't provide enough data to estimate the dispersion reliably, we implement an empirical Bayes strategy for squeezing the tagwise dispersions towards the common dispersion. The amount of shrinkage is determined by the prior weight given to the common dispersion and the precision of the tagwise estimates. The prior can be thought of arising from a number of prior observations, equivalent to `prior.n` tags with common dispersion and the same number of libraries per tag as in the current experiment. The prior number of tags `prior.n` can be set by the user. The precision of the tagwise estimators is roughly proportion to the per-tag degrees of freedom, equal to the number of libraries minus the number of groups or the number of GLM coefficients. We generally recommend choosing `prior.n` so that the total degrees of freedom (`prior.n*df`) associated with the prior is about 20–30, subject to `prior.n` not going below 1. For example, if there are four libraries and two groups, the tagwise degrees of freedom are 2, so we would recommend `prior.n=10`. This is an empirical rule of thumb borne out of experience with a number of datasets.

The qCML method calculates the likelihood conditioning on the total counts for each tag, and uses pseudo counts after adjusted for library sizes. Given a table counts or a `DGEList` object, the qCML common dispersion can be calculated using the `estimateCommonDisp()` function, and the qCML tagwise dispersions can be calculated using the `estimateTagwiseDisp()` function.

However, the qCML method is only applicable on dataset with single factor design since it fails to take into account the effects from multiple factors in a more complicated experiment. Therefore, the qCML method (i.e. the `estimateCommonDisp()` and `estimateTagwiseDisp()` function) is recommended for a study with single factor. When experiment has more than one factor involved, we need to seek a new way of estimating dispersions.

Here is a simple example in estimating dispersions using the qCML method. Given a `DGEList` object `D`, we estimate the dispersions using the following commands.

To estimate common dispersion:

```
D <- estimateCommonDisp(D)
```

To estimate tagwise dispersions:

```
D <- estimateTagwiseDisp(D)
```

Note that common dispersion needs to be estimated before estimating tagwise dispersions.

For more detailed examples, see the case studies in section 11 (Zhang's data), section 12 ('t Hoen's data), section 13 (Li's data), section 14 (Tuch's data) and section 15 (arabidopsis RNA-Seq data).

7.2 Testing for DE genes

For all the Next-Gen squencing data analyses we consider here, people are most interested in finding differentially expressed genes/tags between two (or more) groups. Once negative binomial models are fitted and dispersion estimates are obtained, we can proceed with testing procedures for determing differential expression using the exact test.

The exact test is based on the qCML methods. Knowing the conditional distribution for the sum of counts in a group, we can compute exact p-values by summing over all sums of counts that have a probability less than the probability under the null hypothesis of the observed sum of counts. The exact test for the negative binomial distribution has strong parallels with Fisher's exact test.

As we dicussed in the previous section, the exact test is only applicable to experiments with a single factor. The testing can be done by using the function `exactTest()`, and the function allows both common dispersion and tagwise dispersion approaches. For example:

```
> et <- exactTest(D)
> topTags(et)
```

For more detailed examples, see the case studies in section 11 (Zhang's data), section 12 ('t Hoen's data) and section 13 (Li's data).

8 General experiments (glm functionality)

8.1 Estimating dispersions

For general experiments (with multiple factors), edgeR uses the Cox-Reid profile-adjusted likelihood (CR) method in estimating dispersions. The CR method is derived to overcome the limitations of the qCML method as mentioned above. It takes care of multiple factors by fitting generalized linear models (GLM) with a design matrix.

The CR method is based on the idea of approximate conditional likelihood which reduces to residual maximum likelihood. Given a table counts or a `DGEList` object and the design matrix of the experiment, generalized linear models are fitted. This allows valid estimation of the dispersion, since all systematic sources of variation are accounted for.

The CR method can be used to calculate a common dispersion for all the tags, trended dispersion depending on the tag abundance, or separate dispersions for individual tags. These can be done by calling the functions `estimateGLMCommonDisp()`, `estimateGLMTrendedDisp()` and `estimateGLMTagwiseDisp()`, and it is strongly recommended in multi-factor experiment cases.

Here is a simple example in estimating dispersions using GLM method. Given a `DGEList` object `D` and a design matrix, we estimate the dispersions using the following commands.

To estimate common dispersion:

```
D <- estimateGLMCommonDisp(D, design)
```

To estimate trended dispersions:

```
D <- estimateGLMTrendedDisp(D, design)
```

To estimate tagwise dispersions:

```
D <- estimateGLMTagwiseDisp(D, design)
```

Note that we need to estimate either common dispersion or trended dispersions prior to the estimation of tagwise dispersions. When estimating tagwise dispersions, the empirical Bayes method is applied to squeeze tagwise dispersions towards common dispersion or trended dispersions whichever exists. If both exist, the default is to use the trended dispersions.

For more detailed examples, see the case study in section 14 (Tuch's data).

8.2 Testing for DE genes

For General experiments, once negative binomial models are fitted and dispersion estimates are obtained, we can proceed with testing procedures for determing differential expression using the generalized linear model (GLM) likelihood ratio test.

The GLM likelihood ratio test is based on the idea of fitting negative binomial GLMs with the Cox-Reid dispersion estimates. By doing this, it automatically takes all known sources of varations into account. Therefore, the GLM likelihood ratio test is recommended for experiment with multiple factors.

The testing can be done by using the functions `glmFit()` and `glmLRT()`. Given raw counts, a fixed value for the dispersion parameter and a design matrix, the function `glmFit()` fits the negative binomial GLM for each tag and produces an object of class `DGEGLM` with some new components.

Then this `DGEGLM` object can be passed to the function `glmLRT()` to carry out the likelihood ratio test. User can select coefficient(s) to drop from the full design matrix. This gives the null model against which the full model is compared with in the likelihood ratio test. Tags can then be ranked in order of evidence for differential expression, based on the *p*-value computed for each tag.

As a brief example, consider a situation in which are three treatment groups, each with two replicates, and the researcher wants to make pairwise comparisons between them. A linear model representing the study design can be fitted to the data with commands such as:

```
> group <- factor(c(1,1,2,2,3,3))
> design <- model.matrix(~group)
> fit <- glmFit(y,design,etc)
```

The fit has three parameters. The first is the baseline level of group 1. The second and third are the 2 vs 1 and 3 vs 1 differences.

To compare 2 vs 1:

```
> lrt.2vs1 <- glmLRT(y,fit,coef=2)
> topTags(lrt.2vs1)
```

To compare 3 vs 1:

```
> lrt.3vs1 <- glmLRT(y,fit,coef=3)
```

To compare 3 vs 2:

```
> lrt.3vs2 <- glmLRT(y,fit,contrast=c(0,-1,1))
```

The contrast argument in this case requests a statistical test of the null hypothesis that coefficient3−coefficient2 is equal to zero.

To find genes different between any of the three groups:

```
> lrt <- glmLRT(y,fit,coef=2:3)
> topTags(lrt)
```

For more detailed examples, see the case study in section 14 (Tuch's data) and 15 (arabidopsis RNA-Seq data).

9 What to do if you have no replicates

edgeR is primarily intended for use with data including biological replication. Nevertheless, RNA-Seq and ChIP-Seq are still expensive technologies, so it sometimes happens that only one library can be created for each treatment condition. In these cases there are no replicate libraries from which to estimate biological variability. In this situation, the data analyst is faced with the following choices, none of which are ideal. We do not recommend any of these choices as a satisfactory alternative for biological replication. Rather, they are the best that can be done at the analysis stage, and options 2–4 may be better than assuming that biological variability is absent.

1. Be satisfied with a descriptive analysis, that might include an MDS plot and an analysis of fold changes. Do not attempt a significance analysis. This may be the best advice.

2. Simply pick a reasonable dispersion value, based on your experience with similar data, and use that. Although subjective, this is still more defensible than assuming Poisson variation. Typical values are `dispersion=0.4` for human data, `dispersion=0.1` for data on genetically identical model organisms or `dispersion=0.01` for technical replicates.

3. Remove one or more explanatory factors from the linear model in order to create some residual degrees of freedom. Ideally, this means removing the factors that are least important but, if there is only one factor and only two groups, this may mean removing the entire design matrix or reducing it to a single column for the intercept. If your experiment has several explanatory factors, you could remove the factor with smallest fold changes. If your experiment has several treatment conditions, you could try treating the two most similar conditions as replicates. Estimate the dispersion from this reduced model, then insert these dispersions into the data object containing the full design matrix, then proceed to model fitting and testing with `glmFit` and `glmLRT`. This approach will only be successful if the number of DE genes is relatively small.

   In conjunction with this reduced design matrix, you could try `estimateGLMCommonDisp` with `method="deviance"`, `robust=TRUE` and `subset=NULL`. This is our current best attempt at an automatic method to estimate dispersion without replicates, although it will only give good results when the counts are not too small and the DE genes are a small proportion of the whole. Please understand that this is only our best attempt to return something useable. Reliable estimation of dispersion generally requires replicates.

4. If there exist a sizeable number of control transcripts that should not be DE, the the dispersion could be estimated from them. For example, suppose that `housekeeping` is an index variable identifying housekeeping genes that do not respond to the treatment used in the experiment. First create a copy of the data object with only one treatment group:

```
> d1 <- d
> d1$samples$group <- 1
```

Then estimate the dispersion from the housekeeping genes and all the libraries as one group:

```
> d0 <- estimateCommonDisp(d1[housekeeping,])
```

Then insert this into the full data object and proceed:

```
> d$common.dispersion <- d0$common.dispersion
> et <- exactTest(d)
```

and so on. A reasonably large number of control transcripts is required, at least a few dozen and ideally hundreds.

10 Adjustments for gene length, GC content, mappability and so on

edgeR does not require any adjustment for read count biases related to gene sequence such as gene length, GC content, mappability and so on. While these factors are likely to be important for obtaining a unbiased estimate of the absolute expression level, edgeR does not need absolute expression levels. edgeR is instead concerned with differential expression. Any function of gene sequence will the same gene in the same way each RNA-Seq library, so any gene characteristic such as length or GC content cancels out of genewise comparisons between treatment conditions. Hence we do not recommend adjusting read counts for these or similar factors before attempting an edgeR analysis.

11 Case study: SAGE Profiles of Normal and Tumour Tissue

11.1 Introduction

This section provides a detailed analysis of data from a SAGE experiment to illustrate the data analysis pipeline for edgeR. The data come from a very early study using SAGE technology to analyse gene expression profiles in human cancer cells [Zhang et al., 1997].

11.2 Source of the data

At the time that Zhang et al. [1997] published their paper, no comprehensive study of gene expression in cancer cells had been reported. Zhang et al. [1997] designed a study to address the following issues:

1. How many genes are expressed differentially in tumour versus normal cells?
2. Are the majority of those differences cell-autonomous rather than dependent on the tumour micro-environment?
3. Are most differences cell type-specific or tumour-specific?

They used normal and neoplastic gastro-intestinal tissue as a prototype and analysed global profiles of gene expression in human cancer cells. The researchers derived transcripts from human colorectal (CR) epithelium, CR cancers or pancreatic cancers. The data that we analyse in this case study are Zhang et al. [1997]'s SAGE results for the comparison of expression patterns between normal colon epithelium and primary colon cancer.

They report that the expression profiles revealed that most transcripts were expressed at similar levels, but that 289 transcripts were expressed at significantly different levels (p-value$<$ 0.01) and that 181 of these 289 were decreased in colon tumours as compared with normal colon tissue. Zhang et al. [1997] used Monte Carlo simulation to determine statistical significance. In this case study we will use the edgeR package, based around the negative binomial model, to identify genes differentially expressed in the normal and cancer samples.

11.3 Reading in the data and creating a `DGEList` object

Our first task is to load the edgeR package, read the data into R and organise the data into a `DGEList` object that the functions in the package can recognise. The library size is usually the total sum of all of the counts for a library, and that is how library size is defined in this analysis. The easiest way to construct an appropriate `DGEList` object for these data is described below.

In this case, the tag counts for the four individual libraries are stored in four separate plain text files, `GSM728.txt, GSM729.txt, GSM755.txt` and `GSM756.txt`. In each file, the tag

IDs and counts for each tag are provided in a table. It is best to create a tab-delimited, plain-text 'Targets' file, which, under the headings 'files', 'group' and 'description', gives the filename, the group and a brief description for each sample.

The targets object is produced when the 'Targets.txt' file is read into the R session. This object makes a convenient argument to the function readDGE which reads the tables of counts into our R session, calculates the sizes of the count libraries and produces a DGEList object for use by subsequent functions.

```
> library(edgeR)
> library(limma)
> targets <- read.delim(file = "Targets.txt", stringsAsFactors = FALSE)
> targets

       files group                description
1 GSM728.txt    NC               Normal colon
2 GSM729.txt    NC               Normal colon
3 GSM755.txt    Tu Primary colonrectal tumour
4 GSM756.txt    Tu Primary colonrectal tumour

> d <- readDGE(targets, skip = 5, comment.char = "!")
> d

An object of class "DGEList"
$samples
       files group                description lib.size norm.factors
1 GSM728.txt    NC               Normal colon    50179            1
2 GSM729.txt    NC               Normal colon    49593            1
3 GSM755.txt    Tu Primary colonrectal tumour    57686            1
4 GSM756.txt    Tu Primary colonrectal tumour    49064            1

$counts
              1    2    3   4
CCCATCGTCC 1288 1380 1236   0
CCTCCAGCTA  719  458  148 142
CTAAGACTTC  559  558  248 199
GCCCAGGTCA  520  448   22  62
CACCTAATTG  469  472  763 421
57443 more rows ...
```

11.4 Normalization and filtering

We will filter out very lowly expressed tags. Those which have fewer than 5 counts in total cannot possibly achieve statisical significance for DE, so we filter out these tags.

```
> d <- d[rowSums(d$counts) >= 5,]
> dim(d)
```

```
[1] 5012    4
```

```
> d$samples$lib.size
```

```
[1] 50179 49593 57686 49064
```

```
> colSums(d$counts)
```

```
    1     2     3     4 
34970 35764 36940 30325 
```

```
> d$samples$lib.size <- colSums(d$counts)
> d <- calcNormFactors(d)
> d$samples
```

```
        files group                description lib.size norm.factors
1 GSM728.txt    NC                Normal colon    34970        0.976
2 GSM729.txt    NC                Normal colon    35764        0.965
3 GSM755.txt    Tu Primary colonrectal tumour    36940        0.971
4 GSM756.txt    Tu Primary colonrectal tumour    30325        1.094
```

We see that the vast majority of tags sequenced in this experiment are detected at very low levels. This filtering step reduces the dataset from over 50,000 tags to just over 5000. While this may seem drastic, there is simply no information for DE in the tags we have filtered out. Nevertheless, the filtering reduces the library sizes (total counts in each library) by about 30%.

In the output above we also show the application of TMM normalization to these data using the function calcNormFactors. The normalization factors here are all very close to one, which indicates that the four libraries are very similar in composition.

This DGEList is now ready to be passed to the functions that do the calculations to determine differential expression levels for the genes. Note that when we 'see' the DGEList object d, the counts for just the first five genes in the table are shown, as well as the library sizes and groups for the samples.

11.5 Estimating the dispersion

The first major step in the analysis of DGE data using the NB model is to estimate the dispersion parameter for each tag. The most straight-forward analysis of DGE data uses the common dispersion estimate as the dispersion for all tags. It is simpler than estimating the dispersion separately for each tag, but it does not provides results as appropriate as using the tagwise dispersions.

Estimating the common dispersion is done using the function estimateCommonDisp.

```
> d <- estimateCommonDisp(d)
> names(d)
```

```
[1] "samples"          "counts"           "common.dispersion"
[4] "pseudo.alt"       "conc"             "common.lib.size"
```

The output of `estimateCommonDisp` is a DGEList object with several new elements. The element `common.dispersion`, as the name suggests, provides the estimate of the common dispersion. The element `genes` contains the information about gene/tag identifiers.

Under the negative binomial model, the square root of the common dispersion gives the coefficient of variation of biological variation. Here, as seen in the code below, the coefficient of variation of biological variation is found to be 0.44. We also note that a common dispersion estimate of 0.2 means that there is a lot more variability in the data that can be accounted for by the Poisson model—if a tag has just 200 counts on average in each library, then the estimate of the tag's variance under the NB model is over 40 times greater than it would be under the Poisson model.

```
> d$common.dispersion
```

```
[1] 0.197
```

```
> sqrt(d$common.dispersion)
```

```
[1] 0.444
```

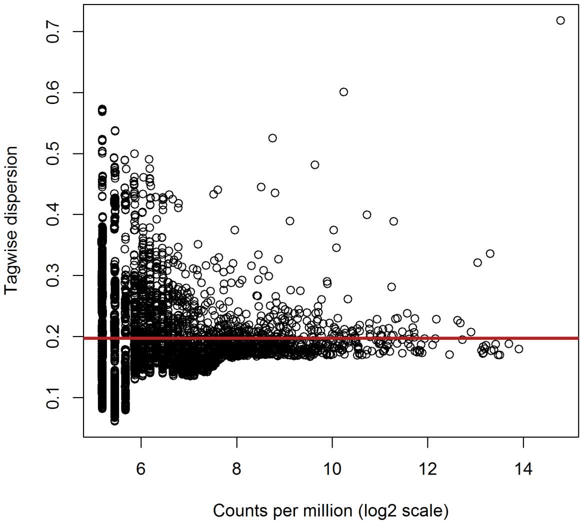
An extension to simply using the common dispersion for each tag is to estimate the dispersion separately for each tag, while 'squeezing' these estimates towards the common dispersion estimate in order to improve inference by sharing information between tags. This type of analysis can also be carried out in few steps using the edgeR package.

The function `estimateTagwiseDisp` produces a DGEList object that contains all of the elements present in the object produced by `estimateCommonDisp`, and the tagwise dispersion estimates (`d$tagwise.dispersion`).

```
> d <- estimateTagwiseDisp(d)
```

The distribution of the tagwise dispersion estimates can be viewed in the following figure, which plots the tagwise dispersion estimates against counts per million (i.e. tag abundance). We see that there are more tags at the lower-abundance end of the spectrum and that these tags tend to be more variable in terms of tagwise dispersion estimates (many tags have dispersion estimates much higher or lower than the common dispersion). For higher abundance levels, tagwise estimates tend to be quite close to the common estimate, apart from those handful of tags which are markedly more variable than the others.

```
> plot(log2(1e06*d$conc$conc.common), d$tagwise.dispersion,
+      xlab="Counts per million (log2 scale)", ylab="Tagwise dispersion")
> abline(h=d$common.dispersion, col="firebrick", lwd=3)
```



11.6 Differential expression

Once we have the estimates of the dispersion, we can proceed with testing procedures for determining differential expression. The edgeR package uses an exact test for the negative binomial distribution, which has strong parallels with Fisher's exact test, to compute exact p-values that can be used to assess differential expression. The function `exactTest` allows the user to conduct the NB exact test for pairwise comparisons of groups. Here there are only two groups, so the pair need not be specified—the function by default compares the two groups present.

```
> et <- exactTest(d)
> names(et)

[1] "table"      "comparison" "genes"

> names(et$table)

[1] "logFC"  "logCPM" "PValue"
```

The object produced by `exactTest` contains three elements: `table`, `comparison` and `genes`. The element `de.com$comparison` contains a vector giving the names of the two groups compared. The table `de.com$table` contains the elements `logFC`, which gives the log-fold change difference for the counts between the groups, `logCPM`, which gives the log counts per million for a tag across the two groups being compared and `PValue` gives the exact p-values computed.

The results of the NB exact test can be accessed conveniently using the `topTags` function applied to the object produced by `exactTest`. The user can specify the number, `n`, of tags for which they would like to see the differential expression information, ranked by p-value (default) or fold change. As the same test is conducted for many thousands of tags, adjusting the p-values for multiple testing is recommended. The desired adjustment method can be supplied by the user, with the default method being Benjamini and Hochberg's approach for controlling the false discovery rate (FDR) [Benjamini and Hochberg, 1995]. The table below shows the top 10 DE genes ranked by p-value.

The output below shows that the edgeR package identifies a good deal of differential expression between the normal colon cell group and the primary CR cancer cell group. The top DE genes are given very small p-values, even after adjusting for multiple testing. Furthermore, all of the top genes have a large fold change, indicating that these genes are more likely to be biologically meaningful. A Gene Ontology analysis could be carried out using the list of top genes and p-values provided by `topTags` in order to obtain more systematic and functional information about the differentially expressed genes.

```
> topTags(et)

Comparison of groups:  Tu-NC
           logFC logCPM   PValue      FDR
AGCTGTTCCC 12.14  13.04 1.07e-13 5.36e-10
CTTGGGTTTT  8.88   9.77 9.03e-09 2.26e-05
GTCATCACCA -7.79   8.65 1.82e-07 2.09e-04
TACAAAATCG  8.13   9.02 2.00e-07 2.09e-04
TCACCGGTCA -4.11  10.54 2.08e-07 2.09e-04
TAAATTGCAA -4.14  10.28 5.89e-07 4.92e-04
TAATTTTTGC  5.54   8.73 1.13e-06 7.62e-04
ATTTCAAGAT -5.50   8.70 1.32e-06 7.62e-04
GTGCGCTGAG  7.36   8.22 1.37e-06 7.62e-04
CTTGACATAC -7.26   8.11 2.11e-06 1.02e-03
```

The table below shows the counts per million for the genes that edgeR has identified as the most differentially expressed. For these genes there seems to be very large differences between the groups, suggesting that the DE genes identified are truly differentially expressed, and not false positives.

```
> detags <- rownames(topTags(et)$table)
> cpm(d)[detags, ]

              1    2      3     4
AGCTGTTCCC    0    0 3221.4 33339
CTTGGGTTTT    0    0  568.5  3199
GTCATCACCA 1001  559    0.0     0
TACAAAATCG    0    0  379.0  1847
TCACCGGTCA 3374 2097  162.4   165
TAAATTGCAA 2945 1650   81.2   198
TAATTTTTGC    0   28 1001.6   692
ATTTCAAGAT 1001  587    0.0    33
GTGCGCTGAG    0    0  487.3   758
CTTGACATAC  515  559    0.0     0
```

Zhang et al. [1997] identified 289 genes as significantly differentially expressed with p-values less than 0.01. We can look at the genes that are given an exact p-value less than 0.01 by edgeR before adjusting for multiple testing, and less than 0.05 after adjustment.

We see in the output below that 222 genes are significantly differentially expressed according to edgeR when using the tagwise dispersion estimates. Of those genes, 87 are up-regulated in the cancer cells compared with the normal cells and 135 are down-regulated in the cancer cells compared with normal cells. These proportions of up- and down-regulated tags are very similar to those found by Zhang et al. [1997].

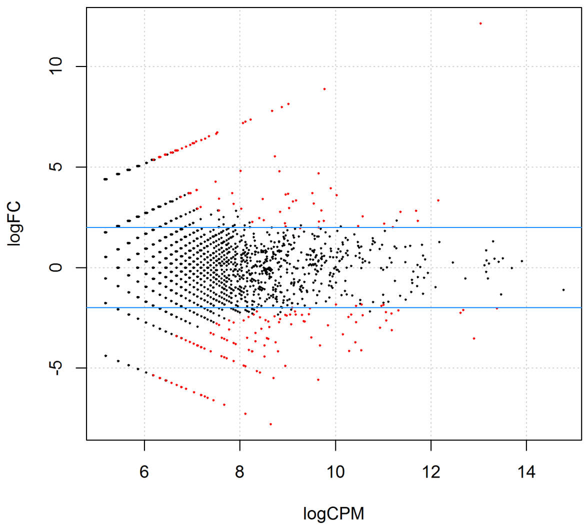
```
> sum(et$table$PValue < 0.01)

[1] 222

> top <- topTags(et, n = 222)
> sum(top$table$logFC > 0)

[1] 87

> sum(top$table$logFC < 0)

[1] 135
```

Furthermore, we see below that 83 tags have a p-value of less than 0.05 after adjusting for multiple testing using the Benjamini and Hochberg [1995] method for controlling the FDR, which is strong evidence for differential expression.

```
> summary(decideTestsDGE(et, p.value=0.05))

   [,1]
-1   50
0  4929
1    33
```

The function `plotSmear` can be used to generate a plot of the log-fold change against the log-counts per million for each tag (analogous to an MA-plot in the microarray context). We can easily identify the DE tags and highlight them on the plot. The 222 most differentially expressed tags as identified by edgeR are outlined in red.

```
> detags <- rownames(topTags(et, n = 222)$table)
> plotSmear(et, de.tags=detags)
> abline(h = c(-2, 2), col = "dodgerblue")
```



11.7 Setup

This analysis of Zhang et al. [1997]'s SAGE data was conducted on:

```
> sessionInfo()

R version 2.15.0 alpha (2012-03-06 r58616)
Platform: i386-pc-mingw32/i386 (32-bit)

locale:
[1] LC_COLLATE=English_Australia.1252  LC_CTYPE=English_Australia.1252
[3] LC_MONETARY=English_Australia.1252 LC_NUMERIC=C
[5] LC_TIME=English_Australia.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] edgeR_2.5.15  limma_3.11.17

loaded via a namespace (and not attached):
[1] tools_2.15.0
```

12 Case study: deepSAGE of wild-type vs Dclk1 transgenic mice

12.1 Introduction

This section provides a detailed analysis of data from an experiment seeking to compare deep-sequenced tag-based expression profiling to the microarray platforms that had been previously used to conduct such studies ['t Hoen et al., 2008].

12.2 Source of the data

't Hoen et al. [2008] address both biological and technical questions in their study. The biological question addressed was the identification of transcripts differentially expressed in the hippocampus between wild-type mice and transgenic mice overexpressing a splice variant of the δC-doublecortin-like kinase-1 (*Dclk1*) gene. The splice variant, DCLK-short, makes the kinase constitutively active and causes subtle behavioural phenotypes.

On the technical side, the researchers compare the robustness, resolution and inter-lab portability of Solexa/Illumina's DGE tag profiling approach and five microarray platforms ['t Hoen et al., 2008]. The tag-based gene expression technology in this experiment could be thought of as a hybrid between SAGE and RNA-seq—like SAGE it uses short sequence tags ($\sim$ 17bp) to identify transcripts, but it uses the deep sequencing capabilities of Solexa/Illumina 1G Genome Analyzer to greatly increase the number of tags that can be sequenced. For our purposes we will concentrate solely on the DGE data generated in the experiment.

The RNA samples came from wild-type male C57/BL6j mice and transgenic mice overexpressing DCLK-short with a C57/BL6j background. Tissue samples were collected from four individuals in each of the two groups by dissecting out both hippocampi from each mouse. Total RNA was isolated and extracted from the hippocampus cells and sequence tags were prepared using Illumina's Digital Gene Expression Tag Profiling Kit according to the manufacturer's protocol.

Sequencing was done using Solexa/Illumina's Whole Genome Sequencer. RNA from each biological sample was supplied to an individual lane in one Illumina 1G flowcell. The instrument conducted 18 cycles of base incorporation, then image analysis and basecalling were performed using the Illumina Pipeline. Sorting and counting the unique tags followed, and the raw data (tag sequences and counts) are what we will analyze here. 't Hoen et al. [2008] went on to annotate the tags by mapping them back to the genome. In general, the mapping of tags is an important and highly non-trivial part of a DGE experiment, but we shall not deal with this task in this case study.

The researchers obtained $\sim$ 2.4 million sequence tags per sample, with tag abundance spanning four orders of magnitude. They found the results to be highly reproducible, even across laboratories. Using a dedicated Bayesian model, they found 3179 transcripts to be differentially expressed with a FDR of 8.5%. This is a much higher figure than was found for the microarrays. 't Hoen et al. [2008] conclude that deep-sequencing offers a major advance in robustness, comparability and richness of expression profiling data.

12.3 Reading in the data and creating a DGEList object

Our first task is to load the edgeR package, read the data into R and organise the data into an object that the functions in the package can recognise. In this case, the tag counts for the eight individual libraries are stored in eight separate plain text files, GSM272105.txt, GSM272106.txt, GSM272318.txt, GSM272319.txt, GSM272320.txt, GSM272321.txt, GSM272322.txt and GSM272323.txt.

In each file, the tag IDs and counts for each tag are provided in a table. It is best to create a tab-delimited, plain-text 'Targets' file, which, under the headings 'files', 'group' and 'description', gives the filename, the group and a brief description for each sample.

The targets object is produced when the 'Targets.txt' file is read into the R session. This object makes a convenient argument to the function readDGE which reads the tables of counts into our R session, calculates the sizes of the count libraries and produces a DGEList object for use by subsequent functions. The skip and comment.char arguments to readDGE are passed to in-built R functions for reading in data, such as read.table.

```
> targets <- read.delim("targets.txt", stringsAsFactors = FALSE)
> targets

          files group                          description
1 GSM272105.txt  DCLK transgenic (Dclk1) mouse hippocampus
2 GSM272106.txt    WT          wild-type mouse hippocampus
3 GSM272318.txt  DCLK transgenic (Dclk1) mouse hippocampus
4 GSM272319.txt    WT          wild-type mouse hippocampus
5 GSM272320.txt  DCLK transgenic (Dclk1) mouse hippocampus
6 GSM272321.txt    WT          wild-type mouse hippocampus
7 GSM272322.txt  DCLK transgenic (Dclk1) mouse hippocampus
8 GSM272323.txt    WT          wild-type mouse hippocampus

> d <- readDGE(targets, skip = 5, comment.char = "!")
> colnames(d) <- c("DCLK1","WT1","DCLK2","WT2","DCLK3","WT3","DCLK4","WT4")
```

This DGEList is now ready to be passed to the functions that do the calculations to determine differential expression levels for the genes. Note that when we 'see' the DGEList object d, the counts for just the first five genes in the table are shown, as well as the samples element, which is a data frame constructed from the 'Targets.txt' file and provides the filenames, groups, descriptions and library sizes for the samples.

However, for this dataset, there were over 800 000 unique tags sequenced, most of which have a very small number of counts in total across all libraries. Since it is not possible to achieve statistical significance with fewer than six counts in total for a tag, we filter out tags which have fewer than one count per million in five or more libraries. This reduces our chances of finding spurious DE (that is, DE driven by large counts in only a handful of libraries) and also helps to speed up the calculations we need to perform. The `cpm` function makes it easy to compute counts per million. The subsetting capability of `DGEList` objects makes such filtering very easy to carry out.

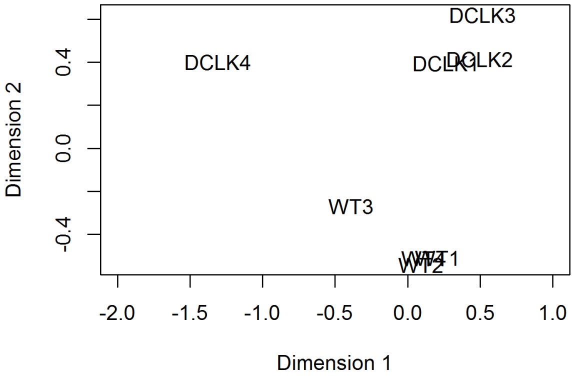
```
> cpm.d <- cpm(d)
> d <- d[rowSums(cpm.d > 1) >= 4, ]
> dim(d)

[1] 44882     8
```

Now the dataset is ready to be analysed for differential expression, with just over 44000 tags remaining with sufficient expression for meaningful DE analysis.

12.4 Data exploration

Before proceeding with the computations for differential expression, it is possible to produce a plot showing the sample relations based on multidimensional scaling. The function `plotMDS` produces an MDS plot for the samples when provided with the `DGEList` object and other usual graphical parameters as arguments, as shown below.

```
> plotMDS(d, xlim=c(-2,1))
```



From the MDS plot, it can be seen that dimension 1 separates the DCLK and WT samples quite nicely.

12.5 Estimating the dispersion

As discussed for the SAGE data, the first major step in the analysis of DGE data using the NB model is to estimate the dispersion parameter for each tag. The most straight-forward analysis of SAGE data uses the common dispersion estimate as the dispersion for all tags. It is simpler than estimating the dispersion separately for each tag, but it does not provides results as appropriate as using the tagwise dispersions.

Estimating the common dispersion is done using the function `estimateCommonDisp`.

```
> d <- estimateCommonDisp(d)
> names(d)
```

```
[1] "samples"           "counts"            "common.dispersion"
[4] "pseudo.alt"        "conc"              "common.lib.size"
```

Here the coefficient of variation of biological variation (square root of the common dispersion) is found to be 0.39. We also note that a common dispersion estimate of 0.15 means that there is a lot more variability in the data that can be accounted for by the Poisson model—if a tag has just 200 counts in total (average of 25 counts per sample), then the estimate of the tag's variance under the NB model is over 10 times greater than it would be under the Poisson model.

```
> d$common.dispersion
```

```
[1] 0.151
```

```
> sqrt(d$common.dispersion)
```

```
[1] 0.389
```

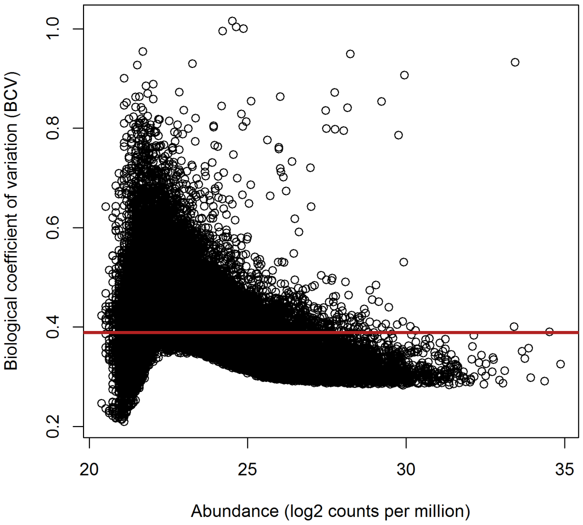
An extension to simply using the common dispersion for each tag is to estimate the dispersion separately for each tag, while 'squeezing' these estimates towards the common dispersion estimate in order to improve inference by sharing information between tags. This type of analysis can also be carried out in few steps using the edgeR package.

The function `estimateTagwiseDisp` produces a DGEList object that contains all of the elements present in the object produced by `estimateCommonDisp`, and the tagwise dispersion estimates (`d$tagwise.dispersion`).

```
> d <- estimateTagwiseDisp(d)
```

As we know, the biological coefficient of variation (BCV) is the square root of the dispersion, the distribution of the BCV can be viewed in the following figure, which plots the BCV against counts per million (i.e. tag abundance). We see that there are more tags at the lower-abundance end of the spectrum and that these tags tend to be more variable in terms of the BCV (many tags have BCV much higher than the common BCV). For higher abundance levels, BCV tends to be quite close to the common estimate, apart from those handful of tags which are markedly more variable than the others.

```
> plotBCV(d)
> abline(h=sqrt(d$common.dispersion), col="firebrick", lwd=3)
```



12.6 Differential expression

Conduct exact conditional tests for differential expression between the mutant and the wild-type:

```
> et <- exactTest(d, pair=c("WT","DCLK"))
```

Top ten differentially expressed tags:

```
> topTags(et)
```

```
Comparison of groups:  DCLK-WT
                  logFC logCPM   PValue      FDR
TCTGTACGCAGTCAGGC  9.40   5.36 9.73e-25 4.37e-20
CATAAGTCACAGAGTCG  9.83   3.48 3.36e-18 7.54e-14
GCTAATAAATGGCAGAT  3.19   5.82 7.62e-16 1.14e-11
ATACTGACATTTCGTAT -4.32   4.33 5.85e-15 6.57e-11
CCAAGAATCTGGTCGTA  3.91   3.54 5.14e-14 4.61e-10
AAAAGAAATCACAGTTG  9.49   3.07 7.26e-14 5.43e-10
TCTGTATGTTCTCGTAT -4.32   4.95 9.55e-14 6.12e-10
CCTATTTTTCTCTCGTA  3.23   6.06 1.54e-13 8.62e-10
TTCCTGAAAATGTGAAG  3.66   3.81 1.92e-13 9.55e-10
CTGCTAAGCAGAAGCAA  3.42   3.80 2.60e-13 1.17e-09
```

The following table shows the individual counts per million for the top ten tags. edgeR chooses tags that both have large fold changes and are consistent between replicates:

```
> detags <- rownames(topTags(et)$table)
> cpm(d)[detags, order(d$samples$group)]
```

```
                  DCLK1  DCLK2  DCLK3  DCLK4    WT1     WT2   WT3   WT4
TCTGTACGCAGTCAGGC  59.58  31.54 178.81  50.68  0.000   0.281  0.00  0.00
CATAAGTCACAGAGTCG  24.95  24.05  23.57  10.75  0.000   0.000  0.00  0.00
GCTAATAAATGGCAGAT 144.11 100.24  53.64 109.03 12.791   8.993  3.39 12.09
ATACTGACATTTCGTAT   1.86   1.56   3.25   1.54 32.121  64.076 13.56 33.10
CCAAGAATCTGGTCGTA  26.07  20.61  19.10  19.96  0.853   1.405  0.00  2.23
AAAAGAAATCACAGTTG  11.54  28.11  17.07   4.61  0.000   0.000  0.00  0.00
TCTGTATGTTCTCGTAT   2.98   3.75   2.84   4.61 26.720 120.002 20.35 56.33
CCTATTTTTCTCTCGTA  83.79 144.59 191.81  50.68 15.634  15.457  3.39 11.77
TTCCTGAAAATGTGAAG  27.56  21.86  34.95  15.36  1.706   2.529  0.00  2.23
CTGCTAAGCAGAAGCAA  28.30  27.48  21.13  23.04  1.990   1.967  0.00  3.50
```

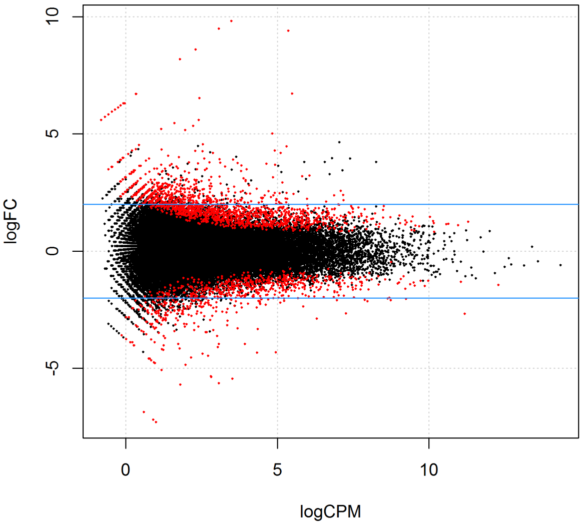
Using their dedicated Bayesian model, 't Hoen et al. [2008] found 3179 transcripts to be differentially expressed at FDR= 8.5%. edgeR finds just slightly fewer tags at the same FDR:

```
> summary(results <- decideTestsDGE(et, p=0.085))
```

```
   [,1]
-1   750
0  42602
1   1530
```

A smear-plot displays all the log-fold changes with the DE genes highlighted:

```
> detags <- rownames(d)[as.logical(results)]
> plotSmear(et, de.tags=detags)
> abline(h = c(-2, 2), col = "dodgerblue")
```



Blue lines indicate 4-fold changes.

12.7 Setup

This analysis of 't Hoen et al. [2008]'s tag-based DGE data was conducted on:

```
> sessionInfo()

R version 2.15.0 alpha (2012-03-06 r58616)
Platform: i386-pc-mingw32/i386 (32-bit)

locale:
[1] LC_COLLATE=English_Australia.1252  LC_CTYPE=English_Australia.1252
[3] LC_MONETARY=English_Australia.1252 LC_NUMERIC=C
[5] LC_TIME=English_Australia.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] edgeR_2.5.15  limma_3.11.17

loaded via a namespace (and not attached):
[1] tools_2.15.0
```

13 Case Study: RNA-seq of Hormone-Treated LNCaP Cells

13.1 Introduction

This case study considers a two-group RNA-seq dataset with relatively low biological variability. It provides a detailed analysis of data from a study by Li et al. [2008] designed to address a range of practical issues in RNA-seq experiments:

1. How many annotated genes are detected in a single cell type?
2. What is the number of tags that is necessary for the analysis of differentially regulated genes under different experimental conditions?
3. To what extent can different mRNA isoforms be detected?
4. How can one quantify alternative splicing by using a single or combination of existing technologies?

Li et al. [2008] attempt to address all of these issues on an androgen-sensitive prostate cancer cell model. We are interested primarily in the second question, and the challenge of identifying differentially regulated genes under different experimental conditions. We will demonstrate the use of the edgeR package for analyzing RNA-seq data for differential gene expression.

13.2 Source of the data

Li et al. [2008] sequenced poly(A)$^+$ RNA from mock-treated or androgen sensitive LNCaP cells (a cell line of human cells commonly used in the field of oncology) on the Illumina 1G Genome Analyzer. The researchers used a double-random priming approach that was capable of generating strand-specific information, although this is not of relevance to our analysis here. The raw RNA-seq data provided by Li et al. consists of 7 'lanes' of 35bp reads. [1] Approximately 10 million sequence tags were generated from both control and hormone-treated cells (treated with DHT), and Li et al. [2008]'s analysis suggests that this tag density is sufficient for quantitative analysis of gene expression.

The 10 million sequenced tags arise from four libraries from control cells and three libraries for hormone-treated cells, giving a total of seven libraries to analyse. From Li et al. [2008] and its companion paper [Li et al., 2006] it is unclear as to whether the treatments are independent or not. The following analysis shows how a quantitative analysis of gene expression, focusing on identifying differentially expressed genes, can be conducted for these seven libraries using edgeR.

[1] The Illumina instrument requires samples to be placed in a 'flow cell' which contains eight 'lanes'—each lane has a sample of cDNA and generates a library of sequence counts for that sample.

13.3 Reading in the data and creating a DGEList object

Our first task is to load the edgeR package and read the data into R. In this case, the tag counts for the libraries are stored in a single table in a plain text file pnas_expression.txt, in which the rows of the table represent tags and the columns represent the different libraries.

To turn the raw RNA-seq data into a table of counts, reads were mapped to the NCBI36 build of the human genome using bowtie, allowing up to two mismatches. Reads which did not map uniquely were discarded. The number of mapped reads that overlapped ENSEMBL gene annotations (version 53) was then counted. In counting reads associated with genes, reads which mapped to non-coding gene regions, such as introns, were included in the count.

Unlike in the other datasets we have look at, counts here are aggregated at the gene, not at the tag, level.

The files object provides the name of the data file, and makes a convenient argument to the function read.delim which reads the table of counts into our R session. We assume that the user can navigate to the directory containing the data file (using, for example, the setwd command in R).

```
> library(edgeR)
> library(limma)
> raw.data <- read.delim("pnas_expression.txt")
> names(raw.data)

[1] "ensembl_ID" "lane1"      "lane2"      "lane3"      "lane4"
[6] "lane5"      "lane6"      "lane8"      "len"
```

The raw data is stored in a table with columns representing the gene names, the counts for the seven libraries and a column giving the length of each gene. The gene length is not currently used by edgeR, but this information could be used in future versions of the package. In the code below, we assign the counts matrix to an object d with the appropriate rownames, define the groups to which the samples belong, and then pass these arguments to DGEList, which calculates the library sizes and constructs a DGEList containing all of the data we require for the analysis.

13.4 Normalization and filtering

We filter out lowly expressed tags and those which are only expressed in a small number of samples. We keep only those tags that have at least one count per million in at least three samples. The counts per million can be computed easily using the cpm function in edgeR.

TMM normalization is applied to this dataset to account for compositional difference between the libraries. As we would hope to see, the normalization factors are very similar within groups and do not differ too greatly between the Control and DHT samples.

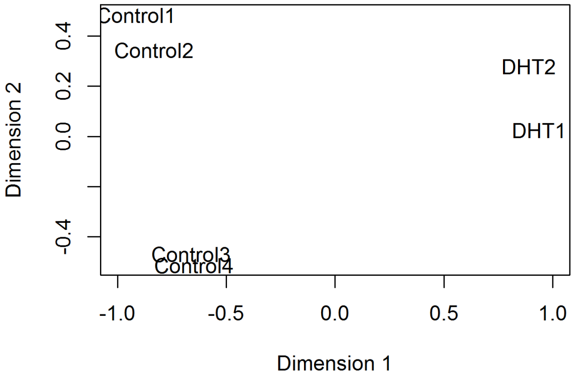
```
> d <- raw.data[, 2:8]
> rownames(d) <- raw.data[, 1]
> group <- c(rep("Control", 4), rep("DHT", 3))
> d <- DGEList(counts = d, group = group)
> dim(d)

[1] 37435     7

> cpm.d <- cpm(d)
> d <- d[ rowSums(cpm.d > 1) >=3, ]
> d <- calcNormFactors(d)
```

This DGEList is now ready to be passed to the functions that do the calculations to determine differential expression levels for the genes.

13.5 Data exploration

Before proceeding with the computations for differential expression, it is possible to produce a plot showing the sample relations based on multidimensional scaling, as demonstrated for the Tag-seq data above. We can produce a multidimensional-scaling (MDS) plot for the Li Data using the command below. An MDS plot can be used to explore similarities or dissimilarities between samples in a visual way.

```
> plotMDS(d, xlim=c(-1,1), labels =
+          c("Control1","Control2","Control3","Control4","DHT1","DHT2","DHT3"))
```



In this plot, Dimension 1 clearly separates the Control from the DHT-treated samples. This shows that the replicates are reasonably similar to each other and that we can expect to find lots of DE genes. Having now investigated some of the relationships between the samples we can proceed to the DE analysis of the data.

13.6 Estimating the dispersion

As discussed for the SAGE data, the first major step in the analysis of DGE data using the NB model is to estimate the dispersion parameter for each tag. The most straight-forward analysis of SAGE data uses the common dispersion estimate as the dispersion for all tags. It is simpler than estimating the dispersion separately for each tag, but it does not provides results as appropriate as using the tagwise dispersions.

```
> d <- estimateCommonDisp(d)
> names(d)

[1] "samples"           "counts"            "all.zeros"
[4] "common.dispersion" "pseudo.alt"        "conc"
[7] "common.lib.size"
```

The output of `estimateCommonDisp` is a DGEList object with several new elements. The element `common.dispersion`, as the name suggests, provides the estimate of the common dispersion. The element `genes` contains the information about gene/tag identifiers.

Here the coefficient of variation of biological variation (square root of the common dispersion) is found to be 0.141. We also note that although a common dispersion estimate of 0.02 may seem 'small', if a tag has just an average of just 200 counts per sample, then the estimate of the tag's variance is 5 times greater than it would be under the Poisson model.

```
> d$common.dispersion

[1] 0.02

> sqrt(d$common.dispersion)

[1] 0.141
```

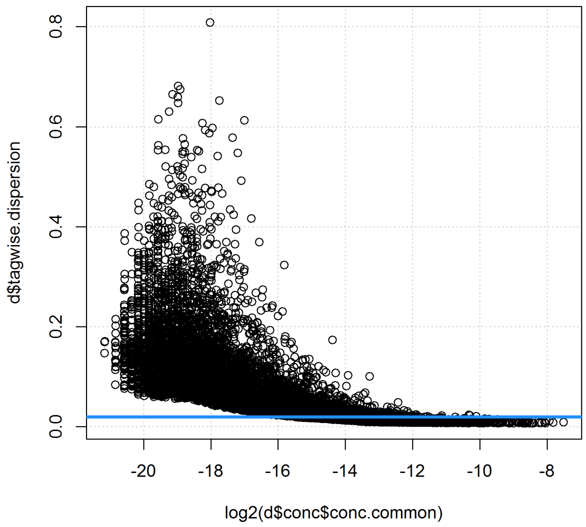
An extension to simply using the common dispersion for each tag is to estimate the dispersion separately for each tag, while 'squeezing' these estimates towards the common dispersion estimate in order to improve inference by sharing information between tags. This type of analysis can also be carried out in few steps using the edgeR package.

The function `estimateTagwiseDisp` produces a DGEList object that contains all of the elements present in the object produced by `estimateCommonDisp`, and the tagwise dispersion estimates (`d$tagwise.dispersion`).

```
> d <- estimateTagwiseDisp(d)
```

The distribution of the tagwise dispersion estimates can be viewed in the following figure, which plots the tagwise dispersion estimates against log-concentration (i.e. tag abundance). Here we have also allowed for a mean-dependent trend on the tagwise dispersion values, which can be inspected in the following figure. As is quite typical for RNA-Seq data, here we see that the tagwise dispersion estimates decrease as the tag abundance increases.

```
> plot(log2(d$conc$conc.common), d$tagwise.dispersion, panel.first=grid())
> abline(h=d$common.dispersion, col="dodgerblue", lwd=3)
```



13.7 Differential expression

Once we have the estimates of the dispersion, we can proceed with testing procedures for determining differential expression. The edgeR package uses an exact test for the negative binomial distribution, which has strong parallels with Fisher's exact test, to compute exact *p*-values that can be used to assess differential expression. The function `exactTest` allows the user to conduct the NB exact test for pairwise comparisons of groups. By default, `exactTest` will use the tagwise dispersion estimates if they are found in the object `d`.

```
> et <- exactTest(d)
```

The output below shows that the edgeR package identifies a huge amount of differential expression between the control group and the DHT-treated group. All of the top genes are up-regulated in the DHT-treated group compared with the control group.

```
> topTags(et)
```

```
Comparison of groups:  DHT-Control
                logFC logCPM    PValue       FDR
ENSG00000151503  5.82   9.71  0.00e+00  0.00e+00
ENSG00000096060  5.00   9.94  0.00e+00  0.00e+00
ENSG00000166451  4.66   8.83 7.19e-229 3.95e-225
ENSG00000127954  8.17   7.20 5.67e-210 2.34e-206
ENSG00000162772  3.32   9.74 1.52e-182 5.00e-179
```

```
ENSG00000113594  4.08   8.03 1.62e-153 4.46e-150
ENSG00000116133  3.26   8.78 4.00e-148 9.42e-145
ENSG00000115648  2.63  11.47 1.30e-139 2.68e-136
ENSG00000123983  3.59   8.58 6.20e-138 1.14e-134
ENSG00000116285  4.22   7.35 6.51e-136 1.07e-132
```

The table below shows the counts per million for the genes that edgeR has identified as the most differentially expressed. For these genes there seems to be very large differences between the groups, suggesting that the DE genes identified are truly differentially expressed, and not false positives.

```
> detags <- rownames(topTags(et)$table)
> cpm(d)[detags, ]
```

```
                lane1 lane2  lane3  lane4 lane5 lane6 lane8
ENSG00000151503  35.8  30.3  33.98  39.71  1814  1875  1795
ENSG00000096060  66.4  68.3  72.81  76.06  2180  2032  2128
ENSG00000166451  41.9  44.9  39.52  38.37   960   902  1068
ENSG00000127954   0.0   0.0   2.08   2.02   333   328   323
ENSG00000162772 175.8 176.3 173.35 204.63  1630  1782  1631
ENSG00000113594  37.8  31.1  39.52  28.94   513   523   613
ENSG00000116133  99.1  92.5 106.78  96.26   895   878   814
ENSG00000115648 960.6 937.0 913.21 905.36  5336  5420  4799
ENSG00000123983  63.4  65.7  65.18  72.70   743   686   921
ENSG00000116285  18.4  24.2  15.95  21.54   354   343   320
```

The decideTestsDGE function provides a useful way to summarize DE results after testing, as shown below.

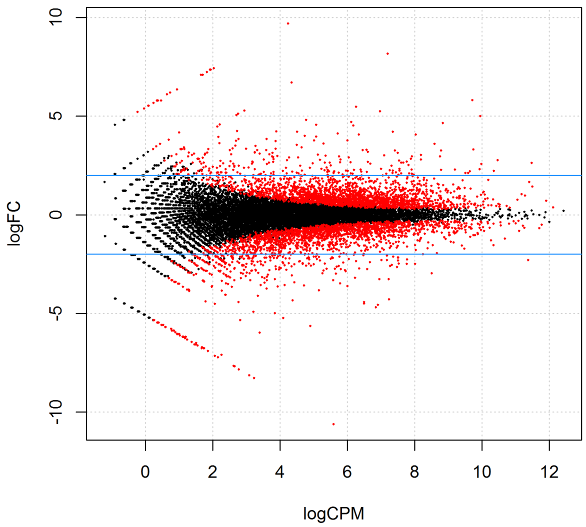
```
> summary(decideTestsDGE(et, p.value=0.05))
```

```
   [,1]
-1  2085
0  12121
1   2288
```

Of the 4373 tags identified as DE using tagwise dispersions, 2085 are up-regulated in DHT-treated cells and 2288 are up-regulated in the control cells.

The function plotSmear can be used to generate a plot of the log-fold change against the log-counts per million for each tag (analogous to an MA-plot in the microarray context). DE tags are highlighted on the plot.

```
> detags <- rownames(topTags(et, n = 4373)$table)
> plotSmear(et, de.tags=detags)
> abline(h = c(-2, 2), col = "dodgerblue")
```



13.8 Setup

The analysis of this section was conducted with:

```
> sessionInfo()

R version 2.15.0 alpha (2012-03-06 r58616)
Platform: i386-pc-mingw32/i386 (32-bit)

locale:
[1] LC_COLLATE=English_Australia.1252  LC_CTYPE=English_Australia.1252
[3] LC_MONETARY=English_Australia.1252 LC_NUMERIC=C
[5] LC_TIME=English_Australia.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] edgeR_2.5.15  limma_3.11.17

loaded via a namespace (and not attached):
[1] tools_2.15.0
```

14 Case study: Oral carcinomas vs matched normal tissue

14.1 Introduction

This section provides a detailed analysis of data from a paired design RNA-seq experiment, featuring oral squamous cell carcinomas and matched normal tissue from three patients [Tuch et al., 2010]. For a paired design, as we discussed before, we have to apply the Cox-Reid (CR) method in estimating dispersions and the GLM method in detecting DE tags.

14.2 Source of the data

The dataset is obtained from the NCBI's Gene Expression Omnibus (GEO) (`http://www.ncbi.nlm.nih.gov/geo/`). It was produced using the Applied Biosystems (AB) SOLiD System 3.0, and is described in Tuch et al. [2010]. The raw reads had been mapped by Tuch et al. [2010] to the UCSC hg18 reference genome. The raw counts, summarised at the level of refSeq transcripts were made available as a supplementary table in their paper. In order to analyse these data in R it is necessary to manipulate the data a little further.

The table that Tuch et al. [2010] provide contains approximately 15000 refSeq transcripts. Many transcripts can map to the same gene, which is not ideal for our analysis in edgeR. It may upset the modeling of the mean-variance relationship for these data if we have several entries for each gene. To get around this problem we have used only the transcipt with the greatest number of exons for each gene, the idea being that this will provide a reasonable summary of the overall expression level for the gene. If the counts were summarised at the exon level, then there are other methods that could be used to find genes with differential isoform expression (or splice variants) from the data.

14.3 Reading in the data and creating a `DGEList` object

Our first task is to load the edgeR package, read the data into R and organise the data into a `DGEList` object that the functions in the package can recognise. The library size is usually the total sum of all of the counts for a library, and that is how library size is defined in this analysis. One way to construct an appropriate `DGEList` object for these data is described below. In this case, the tag counts for the six individual libraries are stored in one table, which is a trimmed version (some irrelevant columns dropped) of the supplementary table from Tuch et al. [2010].

It is usually straight-forward to produce a `DGEList` object from a table of counts, but the task is complicated here because we have many transcripts mapping to the same gene and also the gene symbols provided in the table do not all match exactly to official gene symbols. The commands below show how to ensure that all genes have the official gene symbol (using alias2SymbolTable from the limma package) and that we use only the transcript with the greatest number of exons to represent each gene.

Furthermore, not all of the refSeq IDs provided match the refSeq IDs currently in use—a result of the original study being undertaken several years ago. To avoid potential problems in downstream analysis (particularly in GO or gene set analysis) we retain in our dataset only those transcripts that match to refSeq IDs in the current Entrez database, which is provided by the org.HS.eg.db package from Bioconductor.

The output below shows the commands for manipulating the dataset to produce a neat DGEList object for use by subsequent functions for the DE analysis. We also compute the TMM normalization factors for these libraries in the third last command below.

```
> library(edgeR)
> library(limma)
> rawdata <- read.csv(file="tuch_counts.csv", stringsAsFactors=FALSE)
> library(org.Hs.eg.db)
> rawtable <- rawdata[-c(1,2),]
> refseqid <- as.character(rawtable[,1])
> idfound <- refseqid %in% mappedRkeys(org.Hs.egREFSEQ)
> table(idfound)

idfound
FALSE  TRUE
  367 15301

> rawtable <- rawtable[idfound,]
> genes <- rawtable[,2]
> genes.sym <- alias2SymbolTable(genes, species = "Hs")
> genes <- genes.sym[!is.na(genes.sym)]
> length(genes)

[1] 15262

> nexons <- as.numeric(rawtable[!is.na(genes.sym),3])
> length(nexons)

[1] 15262

> counts <- matrix(as.numeric(unlist(rawtable[!is.na(genes.sym),
+                 -c(1,2,3)])), nrow=sum(!is.na(genes.sym)), ncol=6)
> rownames(counts) <- rawtable[!is.na(genes.sym),1]
> colnames(counts) <- c("N8","T8","N33","T33","N51","T51")
> o <- order(nexons, decreasing=TRUE)
> counts.ord <- counts[o,]
> genes.ord <- genes[o]
> keep <- !duplicated(genes.ord)
> sum(keep)

[1] 10453
```

```
> counts.uniq <- counts.ord[keep,]
> genes.uniq <- genes.ord[keep]
> o2 <- order(genes.uniq)
> d.tuch <- DGEList(counts.uniq[o2,], group=rep(c("normal",
+                           "tumour"),3), genes=genes.uniq[o2])
> d.tuch$samples
```

```
     group lib.size norm.factors
N8  normal  7742608            1
T8  tumour  7126839            1
N33 normal 15581382            1
T33 tumour 13842297            1
N51 normal 20933491            1
T51 tumour 14760103            1
```

This DGEList is now ready to be passed to the functions that do the calculations to determine differential expression levels for the genes.

14.4 Normalization and filtering

TMM normalization is applied to this dataset to account for compositional difference between the libraries.

For this dataset (after our tweaking of it), there are over 10 000 unique tags (genes) sequenced, some of which may have a very small number of counts in total across all libraries. It is not possible to achieve statistical significance with fewer than ten counts in total for a tag, and we also do not want to waste effort finding spurious DE (such as when a gene is only expressed in one library), so we filter out tags with fewer than 1 count per million in four or more libraries—this also helps to speed up the calculations we need to perform. The subsetting capability of DGEList objects makes such filtering very easy to carry out (as shown below). Interestingly, no genes are filtered out for this dataset, indicating that some filtering of low expression transcripts may have been done by Tuch et al. [2010] in producing the table of counts that we have used here.

```
> d.tuch <- calcNormFactors(d.tuch)
> cpm.tuch <- cpm(d.tuch)
> d.tuch <- d.tuch[rowSums(cpm.tuch > 1) >= 2, ]
> nrow(d.tuch)
```

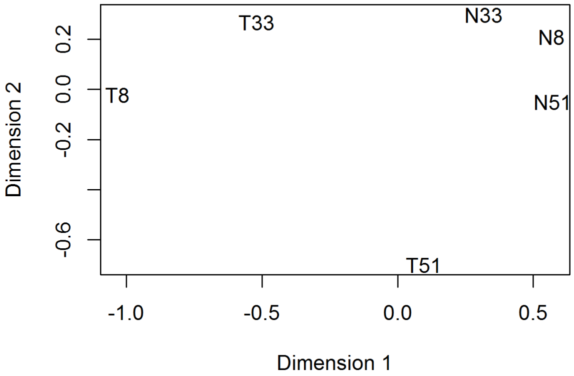
```
[1] 10453
```

Now the dataset is ready to be analysed for differential expression.

14.5 Data exploration

Before proceeding with the computations for differential expression, it is possible to produce a plot showing the sample relations based on multidimensional scaling. The function `plotMDS` produces an MDS plot for the samples when provided with the `DGEList` object, as shown in the figure below.

```
> plotMDS(d.tuch)
```



From the MDS plot, it can be seen that the libraries T51 and T8 (tumour samples from patients 51 and 8 respectively) are most different from the other samples, but we will not remove them from the analysis as we will just be demonstrating the use of edgeR.

14.6 The design matrix

Before we fit negative binomial GLMs, we need to define our design matrix based on the experimental design. Here we want to test for differential expressions between tumour and normal tissues within patients, i.e. adjusting for differences between patients. In statistical terms, this is an additive linear model with patient as the blocking factor. So the full design matrix can be created as follows.

```
> patient <- factor(c(8,8,33,33,51,51))
> design <- model.matrix(~patient+d.tuch$samples$group)
> rownames(design) <- rownames(d.tuch$samples)
> colnames(design)[4] <- "tumour"
> design

    (Intercept) patient33 patient51 tumour
N8            1         0         0      0
T8            1         0         0      1
N33           1         1         0      0
```

```
T33           1         1         0      1
N51           1         0         1      0
T51           1         0         1      1
attr(,"assign")
[1] 0 1 1 2
attr(,"contrasts")
attr(,"contrasts")$patient
[1] "contr.treatment"

attr(,"contrasts")$‘d.tuch$samples$group‘
[1] "contr.treatment"
```

This is the design matrix under the alternative hypothesis (i.e. the difference between the normal tissue and the tumour tissue does exist), and the design matrix under the null hypothesis is just the above matrix without the last column.

14.7 Estimating the dispersion

The first major step in the analysis of DGE data using the NB model is to estimate the dispersion parameter for each tag. Note that this is a paired design experiment, so the dispersion has to be estimated in a way such that both the treatment and the batch effects are taken into account.

The most straight-forward analysis for a paired design experiment uses the common dispersion estimate as the dispersion for all tags. It is simpler than estimating the dispersion separately for each tag, but it does not provides results as appropriate as using the tagwise dispersions.

Estimating the common dispersion is done using the function `estimateGLMCommonDisp()`. Once we have the design matrix, we pass it to the `estimateGLMCommonDisp()` function, together with the `DGEList` object ‘d.tuch’.

```
> d.tuch <- estimateGLMCommonDisp(d.tuch, design)
> names(d.tuch)

[1] "samples"           "counts"            "genes"
[4] "all.zeros"         "common.dispersion"
```

The output of `estimateCRDisp` is a `DGEList` object with several new elements. The element `common.dispersion`, as the name suggests, provides the estimate of the Cox-Reid common dispersion, and `design` gives the design matrix as we defined at the start.

Under the negative binomial model, the square root of the common dispersion gives the coefficient of variation of biological variation. Here the common dispersion is found to be 0.16, so the coefficient of biological variation is around 0.4.

```
> d.tuch$common.dispersion
```

```
[1] 0.16
```

```
> sqrt(d.tuch$common.dispersion)
```

```
[1] 0.4
```

The function `estimateGLMTrendedDisp` in edgeR estimates dispersion values that depend on the overall expression level of the genes. Typically, lowly expressed genes have a higher value for the dispersion parameter than more highly expressed genes. There are a number of possible options for the type of trend that is to be fit for the dispersion parameters. These options are detailed in the help file for `estimateGLMTrendedDisp`.

```
> d.tuch <- estimateGLMTrendedDisp(d.tuch, design)
```

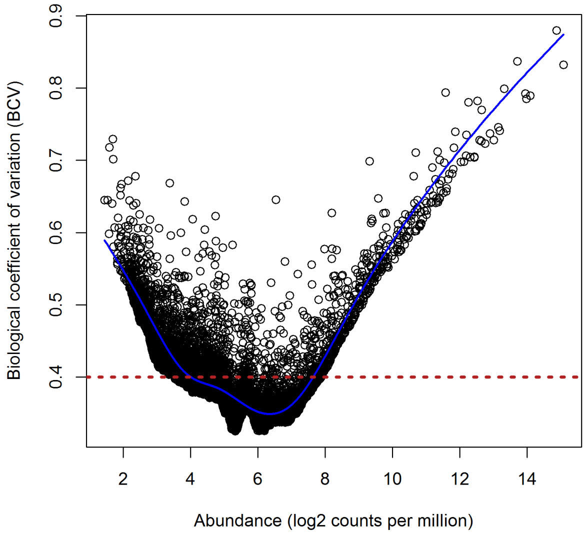
An extension to simply using the common dispersion for each tag is to estimate the dispersion separately for each tag, while 'squeezing' these estimates towards the common dispersion estimate in order to improve inference by sharing information between tags. This type of analysis can also be carried out in few steps using the edgeR package.

The function `estimateGLMTagwiseDisp` produces a DGEList object that contains all of the elements present in the object produced by `estimateGLMCommonDisp`, with an extra element–tagwise dispersions.

```
> d.tuch <- estimateGLMTagwiseDisp(d.tuch, design)
```

A mean-dependent trend on the tagwise dispersion values can be inspected. As is quite typical for RNA-Seq data, here we see that the tagwise dispersion estimates decrease as the tag abundance (log counts per million) increases, as shown in the figure below.

```
> plotBCV(d.tuch)
> abline(h=sqrt(d.tuch$common.dispersion), col="firebrick", lty=3, lwd=3)
```



14.8 Differential expression

Now proceed to determine differentially expressed genes. Fit genewise glms:

```
> fit <- glmFit(d.tuch, design)
```

Make the logFCs more reliable by shrinking them slightly towards zero:

```
> fit$coefficients <- predFC(d.tuch,design) * log(2)
```

Conduct likelihood ratio tests for the group effect and show the top genes.

```
> lrt <- glmLRT(d.tuch, fit)
> topTags(lrt)

Coefficient:  tumour
                 genes logFC logCPM    LR   PValue      FDR
NM_014440        IL36A -6.10   5.43 108.9 1.69e-25 1.77e-21
NM_005609         PYGM -5.46   6.02  96.4 9.44e-23 4.93e-19
NM_001039585     PTGFR -5.18   4.77  93.9 3.25e-22 1.13e-18
NM_182502    TMPRSS11B -7.38   7.67  91.5 1.10e-21 2.88e-18
NM_004320       ATP2A1 -4.62   6.00  85.6 2.23e-20 4.67e-17
NM_004533       MYBPC2 -5.44   6.53  83.9 5.17e-20 9.00e-17
NM_057088         KRT3 -5.80   6.53  82.2 1.21e-19 1.80e-16
NM_002371          MAL -6.86   7.03  81.7 1.56e-19 2.04e-16
NM_001111283      IGF1 -3.99   5.75  78.3 8.64e-19 1.00e-15
NM_001976         ENO3 -5.17   6.35  75.5 3.62e-18 3.78e-15
```

The output shows that the edgeR package identifies a good deal of differential expression between the normal tissue group and the tumour tissue group. The top DE tags are given very small p-values, even after adjusting for multiple testing. Furthermore, all of the top tags have a large fold change, indicating that these tags are more likely to be biologically meaningful.

Here's a closer look at the counts-per-million in individual samples for the top genes:

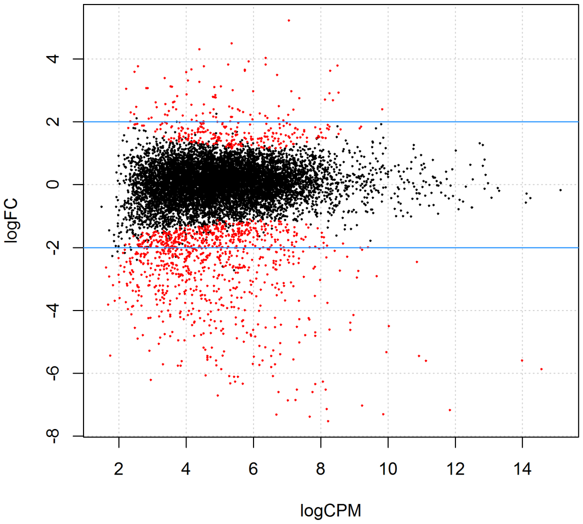
```
> top <- rownames(topTags(lrt)$table)
> cpm(d.tuch)[top,order(d.tuch$samples$group)]

                N8   N33   N51    T8    T33     T51
NM_014440     47.4 117.1  38.3 1.403  3.251  0.0678
NM_005609    180.7  81.3 103.7 3.087  1.156  6.9783
NM_001039585  58.8  18.4  82.9 0.982  0.867  3.1165
NM_182502    334.8 500.9 161.1 0.421 23.190  0.6098
NM_004320    127.6 100.0 109.2 3.508  3.757 10.9078
NM_004533    124.8  31.2 384.3 1.543  0.433 30.9618
NM_057088    138.1 242.2  42.3 0.982 25.863  0.3388
NM_002371    348.3 252.9  83.6 0.421 19.144  0.5420
NM_001111283  59.4  22.0 224.7 3.508  1.878 17.4118
NM_001976    141.0  82.9 231.3 0.561  5.346 15.1083
```

A total of 1278 tags are significantly differentially expressed at 5% FDR. Of those tags, 313 are up-regulated and 965 are down in the tumour group:

```
> summary(decideTestsDGE(lrt))

   [,1]
-1   965
0  9175
1   313
```

The function plotSmear can be used to generate a plot of the log-fold change against the log-counts per million for each tag (analogous to an MA-plot in the microarray context). DE tags are highlighted on the plot.

```
> detags <- rownames(topTags(lrt, n = 1278)$table)
> plotSmear(lrt, de.tags=detags)
> abline(h = c(-2, 2), col = "dodgerblue")
```



14.9 Setup

This analysis of Tuch et al. [2010]'s RNA-seq data was conducted on:

```
> sessionInfo()

R version 2.15.0 alpha (2012-03-06 r58616)
Platform: i386-pc-mingw32/i386 (32-bit)

locale:
```

```
[1] LC_COLLATE=English_Australia.1252  LC_CTYPE=English_Australia.1252
[3] LC_MONETARY=English_Australia.1252 LC_NUMERIC=C
[5] LC_TIME=English_Australia.1252

attached base packages:
[1] splines   stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
[1] org.Hs.eg.db_2.6.4    RSQLite_0.11.1        DBI_0.2-5
[4] AnnotationDbi_1.17.27 Biobase_2.15.4        BiocGenerics_0.1.14
[7] edgeR_2.5.15          limma_3.11.17

loaded via a namespace (and not attached):
[1] IRanges_1.13.30 stats4_2.15.0   tools_2.15.0
```

15 Case study: pathogen inoculated arabidopsis with batch effects

15.1 Introduction

This case study re-analyses arabidopsis thaliana RNA-Seq data described by Cumbie et al. [2011]. Summarized count data is available as a data object in the CRAN package `NBPSeq` comparing ΔhrcC challenged and mock-inoculated samples [Cumbie et al., 2011]. Samples were collected in three batches, and adjustment for batch effects proves to be important.

15.2 RNA samples

Pseudomonas syringae is a bacterium often used to study plant reactions to pathogens. In this experiment, six-week old Arabidopsis plants with inoculated with the ΔhrcC mutant of Pseudomonas syringae, after which total RNA was extracted from leaves. Control plants were inoculated with a mock pathogen.

Three biological replicates of the experiment were conducted at separate times and using independently grown plants and bacteria.

15.3 Sequencing

The six RNA samples were sequenced one per lane an Illumina Genome Analyzer. Reads were aligned and summarized per gene using GENE-counter. The reference genome was derived from the TAIR9 genome release (www.arabidopsis.org).

15.4 Filtering and normalization

Load the data from the NBPSeq package:

```
> library(NBPSeq)
> library(edgeR)
> data(arab)
> head(arab)
```

```
          mock1 mock2 mock3 hrcc1 hrcc2 hrcc3
AT1G01010    35    77    40    46    64    60
AT1G01020    43    45    32    43    39    49
AT1G01030    16    24    26    27    35    20
AT1G01040    72    43    64    66    25    90
AT1G01050    49    78    90    67    45    60
AT1G01060     0    15     2     0    21     8
```

There are two experimental factors, treatment (hrcc vs mock) and the time of each replicate:

```
> Treat <- factor(substring(colnames(arab),1,4))
> Treat <- relevel(Treat, ref="mock")
> Time <- factor(substring(colnames(arab),5,5))
```

There is no purpose in analysing genes that not expressed in either experimental condition. We consider a gene to be expressed at a reasonable level in a sample if it has at least two counts for each million mapped reads in that sample. This cutoff is ad hoc, but serves to require at least 4–6 reads in this case. Since this experiment has three replicates for each condition, a gene should be expressed in at least three samples if it responds to at least one condition. Hence we keep genes with at least one count per million (CPM) in at least three samples:

```
> keep <- rowSums(cpm(arab)>2) >= 3
> arab <- arab[keep, ]
> table(keep)

keep
FALSE  TRUE
 9696 16526
```

Note that the filtering does not use knowledge of what treatment corresponds to each sample, so the filtering does not bias the subrequent differential expression analysis.

Create a DGEList and apply TMM normalization:

```
> y <- DGEList(counts=arab,group=Treat)
> y <- calcNormFactors(y)
> y$samples

      group lib.size norm.factors
mock1  mock  1896802        0.979
mock2  mock  1898690        1.054
mock3  mock  3249396        0.903
hrcc1  hrcc  2119367        1.051
hrcc2  hrcc  1264927        1.096
hrcc3  hrcc  3516253        0.932
```

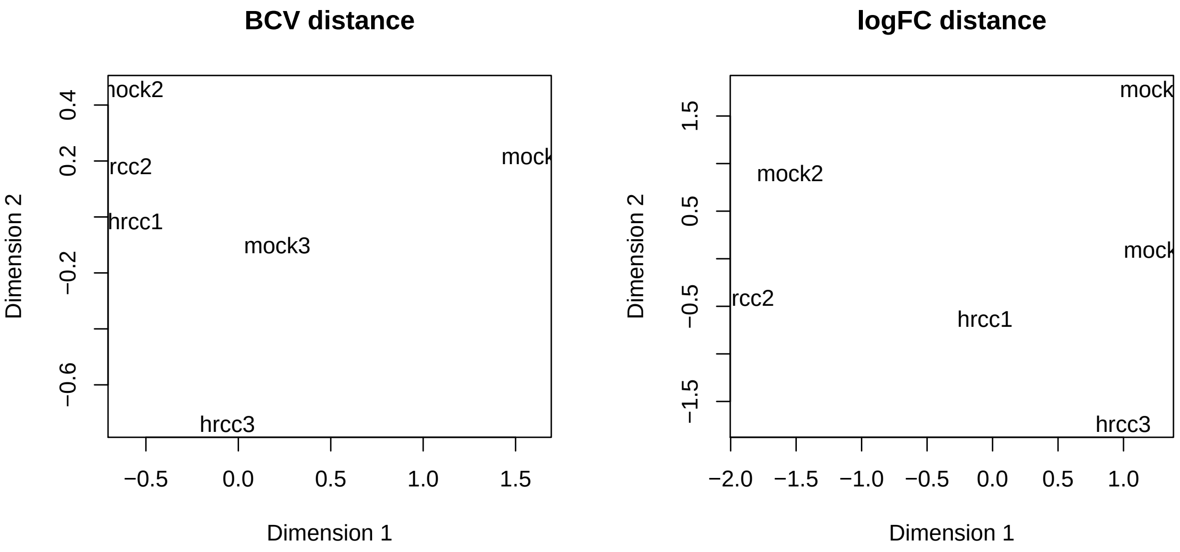
15.5 Data exploration

An MDS plot shows the relative similarities of the six samples. Distances on an MDS plot of a DGEList object correspond to *leading BCV*, the biological coefficient of variation between each pair of samples using the 500 genes with most heterogeneous expression.

```
> plotMDS(y, main="BCV distance")
```

For comparision, we also make an MDS plot with distances defined in terms of shrunk fold changes.

```
> els <- y$samples$lib.size * y$samples$norm.factors
> aug.count <- 2*ncol(arab)*els/sum(els)
> logCPM <- log2( t(t(arab)+aug.count) )
> plotMDS(logCPM, main="logFC distance")
```



The two plots give similar conclusions. Each pair of samples extracted at each time tend to cluster together, suggesting a batch effect. The hrcc treated samples tend to be above the mock samples for each time, suggesting a treatment effect within each time. The two samples at time 1 are less consistent than at times 2 and 3.

To examine further consistency of the three replicates, we compute predictive log2-fold-changes (logFC) for the treatment separately for the three times. The logFC at the three times are positively correlated with one another, as we would hope:

```
> design <- model.matrix(~Time+Time:Treat)
> logFC <- predFC(y,design)
> cor(logFC[,4:6])
```

```
                Time1:Treathrcc Time2:Treathrcc Time3:Treathrcc
Time1:Treathrcc           1.000           0.241           0.309
Time2:Treathrcc           0.241           1.000           0.369
Time3:Treathrcc           0.309           0.369           1.000
```

The correlation is highest between times 2 and 3.

15.6 The design matrix

Before we fit GLMs, we need to define our design matrix based on the experimental design. We want to test for differential expressions between ΔhrcC challenged and mock-inoculated samples within batches, i.e. adjusting for differences between batches. In statistical terms, this is an additive linear model. So the design matrix is created as:

```
> design <- model.matrix(~Time+Treat)
> rownames(design) <- colnames(y)
> design
```

```
      (Intercept) Time2 Time3 Treathrcc
mock1           1     0     0         0
mock2           1     1     0         0
mock3           1     0     1         0
hrcc1           1     0     0         1
hrcc2           1     1     0         1
hrcc3           1     0     1         1
attr(,"assign")
[1] 0 1 1 2
attr(,"contrasts")
attr(,"contrasts")$Time
[1] "contr.treatment"

attr(,"contrasts")$Treat
[1] "contr.treatment"
```

15.7 Estimating the dispersion

The first major step in the analysis of DGE data using the NB model is to estimate the dispersion parameter for each tag. Note that the dispersion has to be estimated in a way such that both the treatment and the batch effects are taken into account.

First estimate the average dispersion over all genes:

```
> y <- estimateGLMCommonDisp(y, design)
> y$common.dispersion
```

```
[1] 0.0706
```

```
> sqrt(y$common.dispersion)
```

```
[1] 0.266
```

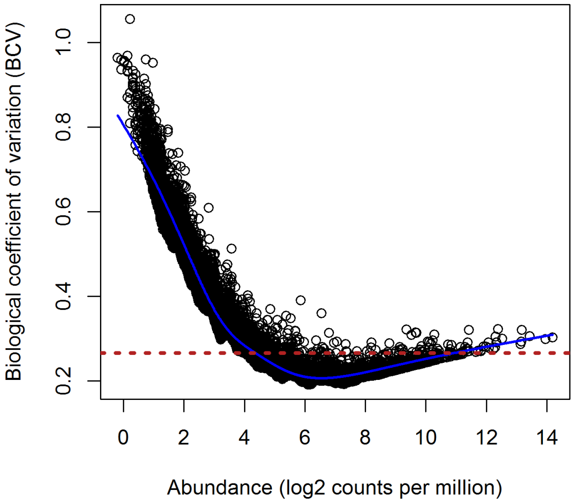
The square root of dispersion is the coefficient of biological variation (BCV). Here the common dispersion is 0.0706, so the BCV is 0.266. The BCV is high, suggesting a very variable response to the treatment, or inconsistency between the replicates.

Now estimate genewise dispersion estimates, allowing for a possible abundance trend:

```
> y <- estimateGLMTrendedDisp(y, design)
> y <- estimateGLMTagwiseDisp(y, design)
```

The genewise show a clear decreasing trend with expression level:

```
> plotBCV(y)
> abline(h=sqrt(y$common.dispersion), col="firebrick", lty=3, lwd=3)
```



15.8 Differential expression

Now proceed to determine differentially expressed genes. Fit genewise glms:

```
> fit <- glmFit(y, design)
```

We make the logFCs more reliable by shrinking them slightly towards zero. This is optional, and does not change the significance results:

```
> fit$coefficients <- predFC(y,design) * log(2)
```

Conduct likelihood ratio tests for the pathogen effect and show the top genes.

```
> lrt <- glmLRT(y, fit)
> topTags(lrt)
```

```
Coefficient:  Treathrcc
          logFC logCPM  LR   PValue      FDR
AT5G48430  6.31   6.71 265 1.19e-59 1.96e-55
AT2G19190  4.50   7.37 228 1.54e-51 1.27e-47
AT3G46280  4.78   8.09 207 4.91e-47 2.71e-43
```

```
AT2G39530  4.34   6.70 197 7.56e-45 3.12e-41
AT2G39380  4.93   5.75 189 4.93e-43 1.63e-39
AT1G51800  3.97   7.70 180 4.63e-41 1.27e-37
AT1G51820  4.33   6.36 169 1.35e-38 3.18e-35
AT2G44370  5.40   5.17 168 1.89e-38 3.91e-35
AT1G51850  5.31   5.39 166 6.21e-38 1.14e-34
AT2G17740  4.21   6.76 165 9.97e-38 1.65e-34
```

Here's a closer look at the individual counts-per-million for the top genes. The top genes are very consistent across the three replicates:

```
> top <- rownames(topTags(lrt)$table)
> cpm(y)[top,order(y$samples$group)]
```

```
          hrcc1 hrcc2 hrcc3 mock1 mock2 mock3
AT5G48430 198.6 344.7 116.6  4.22  4.74  0.00
AT2G19190 358.6 279.1 327.3 16.34 12.64 12.00
AT3G46280 404.4 410.3 765.3 18.45 17.91 16.62
AT2G39530 166.1 210.3 226.7  6.85  9.48 12.00
AT2G39380  96.3  92.5 126.0  2.11  3.16  4.31
AT1G51800 380.8 381.0 432.6 28.47 17.38 27.70
AT1G51820 127.4 171.6 178.3  9.49  7.90  5.54
AT2G44370  59.9  73.5  80.2  2.11  1.05  1.54
AT1G51850  82.1  61.7 101.5  1.05  1.05  3.39
AT2G17740 224.1 224.5 185.1 12.65 18.43  4.92
```

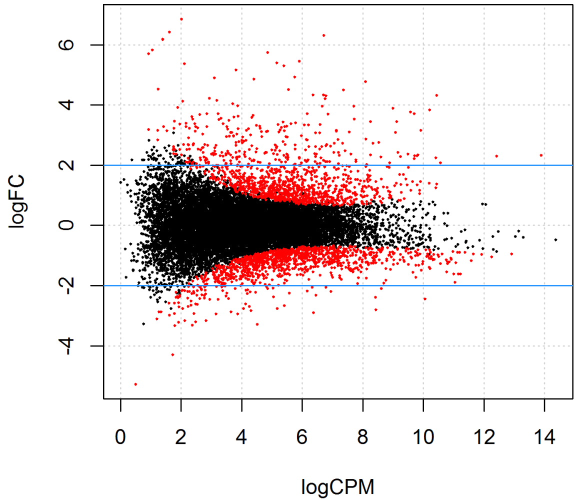
The total number of genes significantly up-regulated or down-regulated at 5% FDR is summarized as follows:

```
> summary(decideTestsDGE(lrt))
```

```
   [,1]
-1  1203
0  14086
1   1237
```

A smear-plot can be used to plot all the logFCs and to highlight the DE genes:

```
> detags <- rownames(topTags(lrt, n=2361)$table)
> plotSmear(lrt, de.tags = detags)
> abline(h = c(-2, 2), col = "dodgerblue")
```



The blue lines indicate 4-fold up or down.

15.9 Setup

This analysis of this arabidopsis thaliana RNA-seq data was conducted on:

```
> sessionInfo()

R version 2.15.0 alpha (2012-03-06 r58616)
Platform: i386-pc-mingw32/i386 (32-bit)

locale:
[1] LC_COLLATE=English_Australia.1252  LC_CTYPE=English_Australia.1252
[3] LC_MONETARY=English_Australia.1252 LC_NUMERIC=C
[5] LC_TIME=English_Australia.1252

attached base packages:
[1] splines   stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
[1] edgeR_2.5.15  limma_3.11.17 NBPSeq_0.1.4  qvalue_1.29.0

loaded via a namespace (and not attached):
[1] tcltk_2.15.0 tools_2.15.0
```

References

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B*, 57:289–300, 1995.

Jason S Cumbie, Jeffrey A Kimbrel, Yanming Di, Daniel W Schafer , Larry J Wilhelm, Samuel E Fox, Christopher M Sullivan, Aron D Curzon, James C Carrington, Todd C Mockler, and Jeff H Chang. Gene-counter: A computational pipeline for the analysis of RNA-Seq data for gene expression differences. *PLoS ONE*, 6(10): e25279, 10 2011. doi: 10.1371/journal.pone.0025279. URL `http://dx.doi.org/10.1371%2Fjournal.pone.0025279`.

H. R Li, J. Wang-Rodriguez, T. M Nair, J. M Yeakley, Y. S Kwon, M. Bibikova, C. Zheng, L. Zhou, K. Zhang, and T. Downs. Two-dimensional transcriptome profiling: identification of messenger RNA isoform signatures in prostate cancer from archived paraffin-embedded cancer specimens. *Cancer Research*, 66(8):4079–4088, 2006.

H. R Li, M. T Lovci, Y-S. Kwon, M. G Rosenfeld, X-D. Fua, and G. W Yeo. Determination of tag density required for digital transcriptome analysis: Application to an androgen-sensitive prostate cancer model. *Proceedings of the National Academy of Sciences of the USA*, 105(51):20179–20184, 2008.

John C Marioni, Christopher E Mason, Shrikant M Mane, Matthew Stephens, and Yoav Gilad. RNA-seq: An assessment of technical reproducibility and comparison with gene expression arrays. *Genome Res*, 18:1509–1517, Jun 2008. doi: 10.1101/gr.079558.108.

Davis J. McCarthy, Yunshun Chen, and Gordon K. Smyth. Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. *Nucleic Acids Research*, 2012. doi: 10.1093/nar/gks042. URL `http://nar.oxfordjournals.org/content/early/2012/02/06/nar.gks042.abstract`.

M. D Robinson and G. K Smyth. Moderated statistical tests for assessing differences in tag abundance. *Bioinformatics*, 23(21):2881–2887, 2007.

M. D Robinson and G. K Smyth. Small-sample estimation of negative binomial dispersion, with applications to SAGE data. *Biostatistics*, 9(2):321–332, 2008.

Mark D Robinson and Alicia Oshlack. A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biology*, 11(3):R25, Mar 2010. doi: 10.1186/gb-2010-11-3-r25. URL `http://genomebiology.com/2010/11/3/R25`.

Mark D Robinson, Davis J McCarthy, and Gordon K Smyth. edgeR: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):

139–40, Jan 2010. doi: 10.1093/bioinformatics/btp616. URL http://bioinformatics.oxfordjournals.org/cgi/content/full/26/1/139.

P. A. C 't Hoen, Y. Ariyurek, H. H Thygesen, E. Vreugdenhil, R. H. A. M Vossen, R. X De Menezes, J. M Boer, G-J. B Van Ommen, and J. T Den Dunnen. Deep sequencing-based expression analysis shows major advances in robustness, resolution and inter-lab portability over five microarray platforms. *Nucleic Acids Research*, 36(21):e141, 2008.

Brian B Tuch, Rebecca R Laborde, Xing Xu, Jian Gu, Christina B Chung, Cinna K Monighetti, Sarah J Stanley, Kerry D Olsen, Jan L Kasperbauer, Eric J Moore, Adam J Broomer, Ruoying Tan, Pius M Brzoska, Matthew W Muller, Asim S Siddiqui, Yan W Asmann, Yongming Sun, Scott Kuersten, Melissa A Barker, Francisco M De La Vega, and David I Smith. Tumor transcriptome sequencing reveals allelic expression imbalances associated with copy number alterations. *PLoS ONE*, 5(2):e9317, Jan 2010. doi: 10.1371/journal.pone.0009317. URL http://www.plosone.org/article/info:doi/10.1371/journal.pone.0009317.

L. Zhang, W. Zhou, V. E Velculescu, S. E Kern, R. H Hruban, S. R Hamilton, B. Vogelstein, and K. W Kinzler. Gene expression profiles in normal and cancer cells. *Science*, 276(5316): 1268–1272, May 1997.