edgeR: differential expression analysis of digital gene expression data

User's Guide

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

First edition 17 September 2008

Last revised 4 May 2012

Contents

Chapter 1

Introduction

1.1 Scope

This guide provides an overview of the Bioconductor package edgeR for differential expression analyses of read counts arising from RNA-Seq, SAGE or similar technologies [Robinson et al., 2010]. The package can be applied to any technology that produces read counts for genomic features. Of particular interest are summaries of short reads from massively parallel 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 (glms), 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.

This guide begins with brief overview of some of the key capabilities of package, and then gives a number of fully worked case studies, from counts to lists of genes.

1.2 Citation

The edgeR package implements statistical methods from the following publications. Please try to cite the appropriate articles if you use the package, as such citation is the main means by which the authors receive credit for their work.

Robinson, MD, and Smyth, GK (2008). Small sample estimation of negative binomial dispersion, with applications to SAGE data. *Biostatistics* 9, 321–332.

> Proposed the idea of sharing information between genes by estimating the negative binomial variance parameter globally across all genes. This made the use of negative binomial models practical for RNA-Seq and SAGE experiments with small to moderate numbers of replicates. Introduced the terminology *dispersion* for the variance parameter. Proposed conditional maximum likelihood for estimating the dispersion, assuming common dispersion across all genes. Developed an exact test for differential expression appropriate for the negative binomially distributed counts. Despite the official publication date, this was the first of the papers to be submitted and accepted for publication.

Robinson, MD, and Smyth, GK (2007). Moderated statistical tests for assessing differences in tag abundance. *Bioinformatics* 23, 2881–2887.

> Introduced empirical Bayes moderated dispersion parameter estimation. This improves on the simple idea fitting a global model to the genewise dispersion parameters.

Robinson, MD, McCarthy, DJ, Smyth, GK (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* 26, 139–140.

> Announcement of the edgeR software package. Introduced the terminology *coefficient of biological variation*.

Robinson MD, Oshlack A (2010). A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biology* 11, R25.

> Introduced the idea of model-based scale normalization of RNA-Seq data. Proposed TMM normalization.

McCarthy, DJ, Chen, Y, Smyth, GK (2012). Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. *Nucleic Acids Research*. Published online 28 January 2012.

> Extended negative binomial differential expression methods to glms, making the methods applicable to general experiments. Introduced the use of Cox-Reid approximate conditional maximum likelihood for estimating the dispersion parameters, and used this for empirical Bayes moderation. Developed fast algorithms for fitting glms to thousands of genes in parallel. Gives a full explanation of the concept of *biological coefficient of variation*.

1.3 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`.

1.4 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 tab-delimited 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.

Chapter 2

Overview of capabilities

2.1 Reading the counts

edgeR works on a table of integer read counts, with rows corresponding to genes or genomic features and columns to independent libraries. The first step in any analysis will usually be to read these counts into an R session. This is straightforward for anyone experienced with R, but can be hurdle for first-timers.

If the count data is contained in a single tab-delimited or comma-separated text file with multiple columns, one for each sample, then the simplest method is usually to read the file into R using one of the standard R read functions such as `read.delim`. See the quick start above, or the case study on LNCaP Cells, or the case study on oral carcinomas later in this guide for examples.

If the counts for different samples are stored in separate files, then the files have to be read separately and collated together. The edgeR function `readDGE` is provided to do this. Files need to contain two columns, one for the counts and one for a gene identifier. See the SAGE and deepSAGE case studies for examples of this.

2.2 The DGEList data class

edgeR stores data in a simple list-based data object called a `DGEList`. This type of object is easy to use because it can be manipulated like any list in R.

The function `readDGE` makes a `DGEList` object directly.

If the table of counts is already available as a matrix or a data.frame, `y` say, then a `DGEList` object can be made by

```
> dge <- DGEList(counts=y)
```

A grouping factor can be added at the same time:

```
> group <- c(1,1,2,2)
> dge <- DGEList(counts=y, group=group)
```

The main components of an `DGEList` object are a matrix `counts` containing the integer counts, a data.frame `samples` containing information about the samples or libraries, and a optional data.frame `genes` containing annotation for the genes or genomic features. The data.frame `samples` contains a column `lib.size` for the library size or sequencing depth for each sample. If not specified by the user, the library sizes will be computed from the column sums of the counts. For classic edgeR the data.frame `samples` must also contain a column `group`, identifying the group membership of each sample.

2.3 Normalization issues for count data

2.3.1 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.

2.3.2 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.

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 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.

In summary, 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.

2.3.3 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, produced by the following code:

```
> 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")
```

In the left panel of Figure 2.1, 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 2.1 simply shows the M and A values after correction. Here, one should find that the bulk of the M-values are centred around 0.

2.4 Negative binomial models

2.4.1 Introduction

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

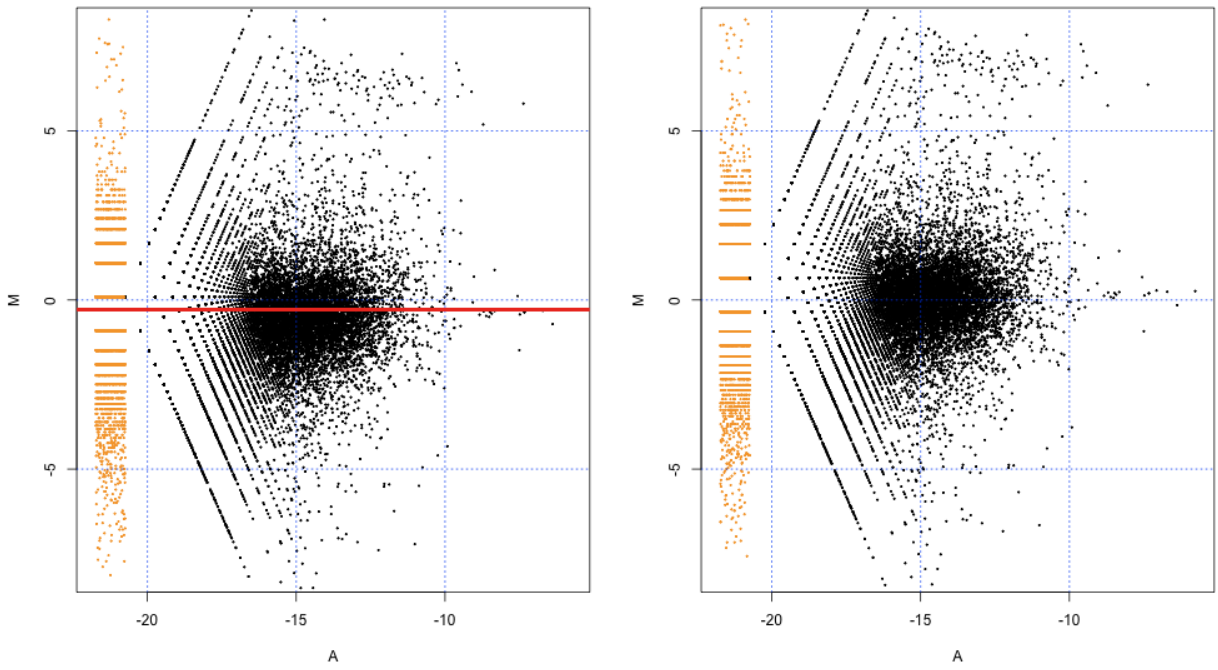


Figure 2.1: Smear plots before (left) and after (right) composition normalization.

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.

2.4.2 Biological coefficient of variation (BCV)

RNA-Seq profiles are formed from n RNA samples. Let π_{gi} be the fraction of all cDNA fragments in the ith sample that originate from gene g. Let G denote the total number of genes, so $\sum_{g=1}^{G} \pi_{gi} = 1$ for each sample. Let $\sqrt{\phi}_g$ denote the coefficient of variation (CV) (standard deviation divided by mean) of π_{gi} between the replicates i. We denote the total number of mapped reads in library i by N_i and the number that map to the gth gene by y_{gi}. Then

$$E(y_{gi}) = \mu_{gi} = N_i \pi_{gi}.$$

Assuming that the count y_{gi} follows a Poisson distribution for repeated sequencing runs of the same RNA sample, a well known formula for the variance of a mixture distribution

implies:

$$\text{var}(y_{gi}) = E_\pi\left[\text{var}(y|\pi)\right] + \text{var}_\pi\left[E(y|\pi)\right] = \mu_{gi} + \phi_g \mu_{gi}^2.$$

Dividing both sides by μ_{gi}^2 gives

$$\text{CV}^2(y_{gi}) = 1/\mu_{gi} + \phi_g.$$

The first term $1/\mu_{gi}$ is the squared CV for the Poisson distribution and the second is the squared CV of the unobserved expression values. The total CV^2 therefore is the technical CV^2 with which π_{gi} is measured plus the biological CV^2 of the true π_{gi}. In this article, we call ϕ_g the dispersion and $\sqrt{\phi_g}$ the biological CV although, strictly speaking, it captures all sources of the inter-library variation between replicates, including perhaps contributions from technical causes such as library preparation as well as true biological variation between samples.

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 the read counts for a particular gene should vary according to a Poisson law Marioni et al. [2008]. If sequencing variation is Poisson, then it can be shown 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.

2.4.3 Estimating BCVs

When a negative binomial model is fitted, we need to estimate the BCV(s) before we carry out the analysis. The BCV, as shown in the previous section, is the square root of the dispersion parameter under the negative binomial model. Hence, it is equivalent to estimating the dispersion(s) of the negative binomial model.

The parallel nature of sequencing data allows some possibilities for borrowing information from the ensemble of genes which can assist in inference about each gene individually. The easiest way to share information between genes is to assume that all genes have the same mean-variance relationship, in other words, the dispersion is the same for all the genes [Robinson and Smyth, 2008]. An extension to this "common dispersion" approach is to put a mean-dependent trend on a parameter in the variance function, so that all genes with the same expected count have the same variance [Anders and Huber, 2010].

However, the truth is that the gene expression levels have non-identical and dependent distribution between genes, which makes the above assumptions too naive. A more general approach that allows genewise variance functions with empirical Bayes shrinkage was introduced several years ago [Robinson and Smyth, 2007] and has recently been extended to generalized linear models and thus more complex experimental designs [McCarthy et al., 2012]. Only when using tagwise dispersion will genes that are consistent between replicates be ranked more highly than genes that are not. It has been seen in many RNA-Seq datasets that allowing gene-specific dispersion is necessary in order that differential expression is not driven by outliers. Therefore, the tagwise dispersions are strongly recommended in model fitting and testing for differential expression.

In edgeR, we first estimate a common dispersion for all the tags and then apply 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. 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 default behavior of the edgeR is to set the prior degrees of freedom to 20.

2.5 Pairwise comparisons between two or more groups (classic)

2.5.1 Estimating dispersions

edgeR uses the quantile-adjusted conditional maximum likelihood (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 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 3.1 (Zhang's data), section 3.2 ('t Hoen's data) and section 3.3 (Li's data).

2.5.2 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 3.1 (Zhang's data), section 3.2 ('t Hoen's data) and section 3.3 (Li's data).

2.6 More complex experiments (glm functionality)

2.6.1 Generalized linear models

Generalized linear models (GLMs) are an extension of classical linear models to nonnormally distributed response data Nelder and Wedderburn [1972], McCullagh and Nelder [1989]. GLMs specify probability distributions according to their mean-variance relationship, for example the quadratic mean-variance relationship specified above for read counts. Assuming that an estimate is available for ϕ_g, so the variance can be evaluated for any value of μ_{gi}, GLM theory can be used to fit a log-linear model

$$\log \mu_{gi} = \mathbf{x}_i^T \boldsymbol{\beta}_g + \log N_i$$

for each gene Lu et al. [2005], Bullard et al. [2010]. Here $\mathbf{x}_i$ is a vector of covariates that specifies the treatment conditions applied to RNA sample i, and $\boldsymbol{\beta}_g$ is a vector of regression coefficients by which the covariate effects are mediated for gene g. The quadratic variance function specifies the negative binomial GLM distributional family. The use of the negative binomial distribution is equivalent to treating the π_{gi} as gamma distributed.

2.6.2 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 3.4 (Tuch's data).

2.6.3 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 3.4 (Tuch's data) and 3.5 (arabidopsis RNA-Seq data).

2.7 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.

Chapter 3

Case studies

3.1 SAGE profiles of normal and tumour tissue

3.1.1 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].

Zhang et al. [1997] examined human colorectal and pancreatic cancer tumor tissue. In this case study, we analyse the data comparing primary colon tumor tissue with normal colon epithelium cells. Two tumor and two normal RNA samples were available from different individuals.

3.1.2 Reading the data

The tag counts for the four individual libraries are stored in four separate plain text files obtained from the GEO repository:

```
> dir()
[1] "GSM728.txt"  "GSM729.txt"  "GSM755.txt"  "GSM756.txt"  "targets.txt"
```

In each file, the tag IDs and counts for each tag are provided in a table.

The file `targets.txt` gives the filename, the group and a brief description for each sample:

```
> targets <- readTargets()
> 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
```

This makes a convenient argument to the function `readDGE`, which reads the tables of counts, calculates the sizes of the count libraries and produces a `DGEList` object for use by subsequent functions. The `skip` and `comment.char` arguments are used to ignore comment lines:

```
> d <- readDGE(targets, skip=5, comment.char = "!")
> d$samples
       files group                description lib.size norm.factors
1 GSM728.txt    NC               Normal colon    50179            1
2 GSM729.txt    NC               Normal colon    49593            1
3 GSM755.txt    Tu Primary colonrectal tumour    57686            1
4 GSM756.txt    Tu Primary colonrectal tumour    49064            1
> head(d$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
CCTGTAATCC  448  229  459 374
> summary(d$counts)
       1               2               3               4
 Min.   :   0   Min.   :   0   Min.   :   0   Min.   :   0
 1st Qu.:   0   1st Qu.:   0   1st Qu.:   0   1st Qu.:   0
 Median :   0   Median :   0   Median :   0   Median :   0
 Mean   :   1   Mean   :   1   Mean   :   1   Mean   :   1
 3rd Qu.:   1   3rd Qu.:   1   3rd Qu.:   1   3rd Qu.:   1
 Max.   :1288   Max.   :1380   Max.   :1236   Max.   :1011
```

There are 57448 unique tags:

```
> dim(d)
[1] 57448     4
```

3.1.3 Filter low expression tags

The number of unique tags is greater than the total number of reads in each library, so the average number of reads per tag per sample is less than one. We will filter out tags with very low counts. We want to keep tags that are expressed in at least one normal or tumor cells. Since there are two replicate samples in each group, we keep tags that are expressed at a reasonable level in at least two samples. Our expession cutoff is 100 counts per million (cpm). For the library sizes here, 100 cpm corresponds to a read count of about 5:

```
> keep <- rowSums(cpm(d)>100) >= 2
> d <- d[keep,]
> dim(d)
[1] 1233    4
```

This reduces the dataset to around 1200 tags. For the filtered tags, there is very little power to detect differential expression, so little information is lost by filtering.

After filtering, it is a good idea to reset the library sizes:

```
> d$samples$lib.size <- colSums(d$counts)
> d$samples
       files group                description lib.size norm.factors
1 GSM728.txt    NC               Normal colon    27012            1
2 GSM729.txt    NC               Normal colon    27735            1
3 GSM755.txt    Tu Primary colonrectal tumour    28696            1
4 GSM756.txt    Tu Primary colonrectal tumour    22461            1
```

3.1.4 Normalization

Apply TMM normalization:

```
> d <- calcNormFactors(d)
> d$samples
       files group                description lib.size norm.factors
1 GSM728.txt    NC                Normal colon    27012        0.989
2 GSM729.txt    NC                Normal colon    27735        1.005
3 GSM755.txt    Tu Primary colonrectal tumour    28696        0.906
4 GSM756.txt    Tu Primary colonrectal tumour    22461        1.110
```

The normalization factors here are all very close to one, indicating 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.

3.1.5 Estimating the dispersions

The first major step in the analysis of DGE data using the NB model is to estimate the dispersion parameter for each tag, a measure of the degree inter-library variation for that tag. Estimating the common dispersion gives an idea of overall variability across the genome for this dataset:

```
> d <- estimateCommonDisp(d, verbose=TRUE)
Disp = 0.173 , BCV = 0.416
```

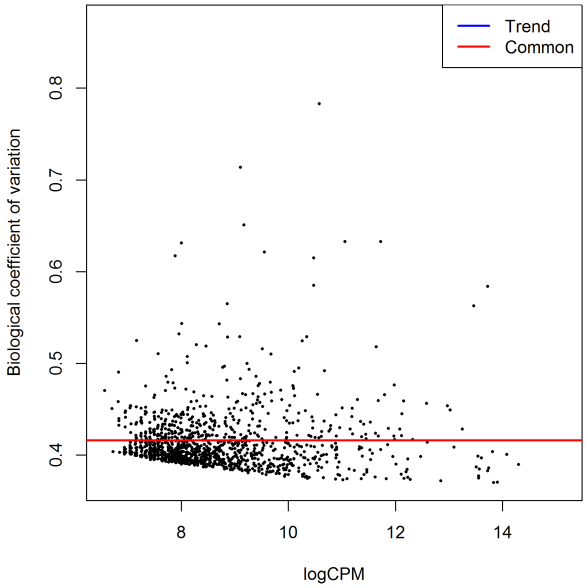
The square root of the common dispersion gives the coefficient of variation of biological variation (BCV). Here the BCV is 41%. This is a relatively large value, but not untypical for observational studies on human tumor tissue where the replicates are independent tumors or individuals.

For routine differential expresion analysis, we use empirical Bayes tagwise dispersions. For SAGE date, no abundance-dispersion trend is usually necessary:

```
> d <- estimateTagwiseDisp(d, trend="none")
```

`plotBCV()` plots the tagwise dispersions against log2-CPM:

```
> plotBCV(d, cex=0.4)
```



3.1.6 Differential expression

Once the dispersions are estimated, we can proceed with testing procedures for determining differential expression. The function `exactTest` conducts tagwise tests using the exact negative binomial test proposed by Robinson and Smyth [2008]. The test results for the `n` most significant tags are conveniently displayed by the `topTags` function:

```
> et <- exactTest(d)
> topTags(et, n=20)
Comparison of groups:  Tu-NC
           logFC logCPM   PValue      FDR
AGCTGTTCCC 12.19  13.46 6.55e-14 8.08e-11
CTTGGGTTTT  8.94  10.19 3.57e-09 2.20e-06
TCACCGGTCA -4.00  10.88 5.06e-08 2.08e-05
TACAAAATCG  8.19   9.43 8.18e-08 2.15e-05
GTCATCACCA -7.74   9.00 8.72e-08 2.15e-05
TAATTTTTGC  5.63   9.16 2.71e-07 5.58e-05
TAAATTGCAA -4.03  10.63 3.40e-07 5.99e-05
GTGCGCTGAG  7.42   8.64 5.25e-07 7.98e-05
GGCTTTAGGG  3.44  12.59 5.82e-07 7.98e-05
ATTTCAAGAT -5.40   9.05 7.37e-07 9.08e-05
GCCCAGGTCA -3.42  13.25 1.15e-06 1.19e-04
GTGTGTTTGT  7.31   8.53 1.18e-06 1.19e-04
CGCGTCACTA  4.78  10.09 1.25e-06 1.19e-04
CTTGACATAC -7.21   8.46 1.44e-06 1.27e-04
GACCAGTGGC -4.78   9.29 1.57e-06 1.29e-04
CCAGTCCGCC  7.84   9.09 2.22e-06 1.71e-04
GGAACTGTGA -3.62  10.76 3.36e-06 2.44e-04
```

```
CCTTCAAATC -5.12   8.77 3.57e-06 2.45e-04
GCAACAACAC  3.81   9.94 3.78e-06 2.45e-04
GATGACCCCC -3.37   9.84 7.70e-06 4.75e-04
```

By default, Benjamini and Hochberg's algorithm is used to control the false discovery rate (FDR) [Benjamini and Hochberg, 1995].

The table below shows the counts per million for the tags that edgeR has identified as the most differentially expressed. There are pronounced differences between the groups:

```
> detags <- rownames(topTags(et, n=20))
> cpm(d)[detags,]
               1       2       3       4
AGCTGTTCCC     0     0.0  4146.9 45011.4
CTTGGGTTTT     0     0.0   731.8  4318.6
TCACCGGTCA  4368  2704.2   209.1   222.6
TACAAAATCG     0     0.0   487.9  2493.2
GTCATCACCA  1296   721.1     0.0     0.0
TAATTTTTGC     0    36.1  1289.4   935.0
TAAATTGCAA  3813  2127.3   104.5   267.1
GTGCGCTGAG     0     0.0   627.3  1024.0
GGCTTTAGGG   777  1298.0 13660.4  8370.1
ATTTCAAGAT  1296   757.2     0.0    44.5
GCCCAGGTCA 19251 16152.9   766.7  2760.3
GTGTGTTTGT     0     0.0   522.7  1024.0
CGCGTCACTA    37   108.2  3066.6   935.0
CTTGACATAC   666   721.1     0.0     0.0
GACCAGTGGC   777  1622.5     0.0    89.0
CCAGTCCGCC     0     0.0   209.1  2181.6
GGAACTGTGA  3332  3028.7    69.7   489.7
CCTTCAAATC  1074   612.9     0.0    44.5
GCAACAACAC   111   144.2  2265.1  1335.6
GATGACCCCC  1555  1766.7   104.5   222.6
```

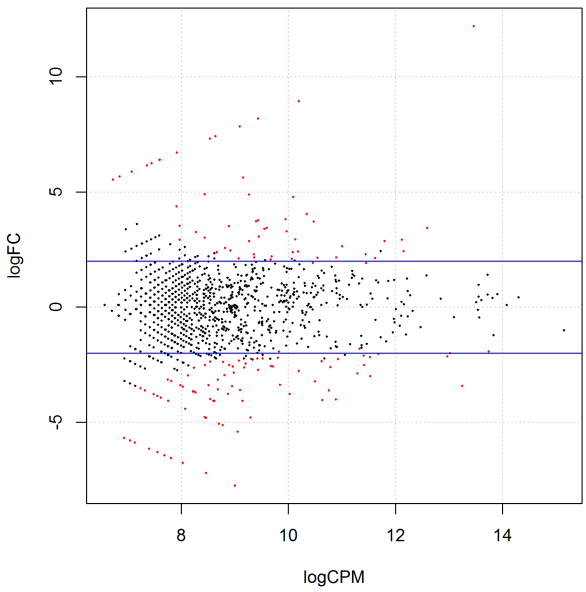
The total number of differentially expressed genes at FDR<0.05 is:

```
> summary(de <- decideTestsDGE(et, p=0.05, adjust="BH"))
   [,1]
-1   87
0  1088
1    58
```

Here the entries for `-1`, `0` and `1` are for down, non-differentially expressed and up tags respectively.

The function `plotSmear` generates a plot of the tagwise log-fold-changes against log-cpm (analogous to an MA-plot for microarray data). DE tags are highlighted on the plot:

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



The horizontal blue lines show 4-fold changes.

3.1.7 Setup

This analysis was conducted on:

```
> sessionInfo()
R version 2.15.0 (2012-03-30)
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_2.7.4  limma_3.13.1

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

3.2 deepSAGE of wild-type vs Dclk1 transgenic mice

3.2.1 Introduction

This section provides a detailed analysis of data from an experiment using deep-sequenced tag-based expression profiling ['t Hoen et al., 2008].

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.

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.

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.

3.2.2 Reading in the data

The tag counts for the eight individual libraries are stored in eight separate plain text files:

```
> dir()
 [1] "GSE10782_Dataset_Summary.txt" "GSM272105.txt"
 [3] "GSM272106.txt"                "GSM272318.txt"
 [5] "GSM272319.txt"                "GSM272320.txt"
 [7] "GSM272321.txt"                "GSM272322.txt"
 [9] "GSM272323.txt"                "Targets.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.

```
> targets <- read.delim("targets.txt", stringsAsFactors = FALSE)
> targets
          files group                        description
1 GSM272105.txt  DCLK Dclk1 transgenic mouse hippocampus
2 GSM272106.txt    WT        wild-type mouse hippocampus
3 GSM272318.txt  DCLK Dclk1 transgenic mouse hippocampus
4 GSM272319.txt    WT        wild-type mouse hippocampus
5 GSM272320.txt  DCLK Dclk1 transgenic mouse hippocampus
6 GSM272321.txt    WT        wild-type mouse hippocampus
7 GSM272322.txt  DCLK Dclk1 transgenic mouse hippocampus
8 GSM272323.txt    WT        wild-type mouse hippocampus
```

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 are used to skip over comment lines:

```
> d <- readDGE(targets, skip = 5, comment.char = "!")
> colnames(d) <- c("DCLK1","WT1","DCLK2","WT2","DCLK3","WT3","DCLK4","WT4")
> d$samples
              files group                        description lib.size
DCLK1 GSM272105.txt  DCLK Dclk1 transgenic mouse hippocampus  2685418
WT1   GSM272106.txt    WT        wild-type mouse hippocampus  3517977
DCLK2 GSM272318.txt  DCLK Dclk1 transgenic mouse hippocampus  3202246
WT2   GSM272319.txt    WT        wild-type mouse hippocampus  3558260
DCLK3 GSM272320.txt  DCLK Dclk1 transgenic mouse hippocampus  2460753
WT3   GSM272321.txt    WT        wild-type mouse hippocampus   294909
DCLK4 GSM272322.txt  DCLK Dclk1 transgenic mouse hippocampus   651172
WT4   GSM272323.txt    WT        wild-type mouse hippocampus  3142280
      norm.factors
DCLK1            1
WT1              1
DCLK2            1
WT2              1
DCLK3            1
WT3              1
DCLK4            1
WT4              1
> dim(d)
[1] 844316      8
```

3.2.3 Filtering

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. We want to keep tags that are expressed in at least one of wild-type or transgenic mice. In either case, the tag should be expressed

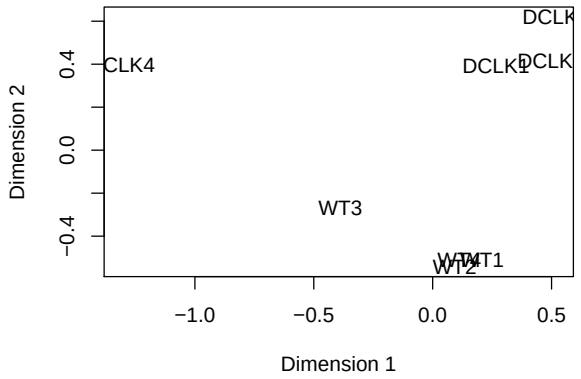
in at least four libraries. We seek tags that achieve one count per million for at least four libraries:

```
> keep <- rowSums(cpm(d) > 1) >= 4
> d <- d[keep,]
> dim(d)
[1] 44882     8
```

3.2.4 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:

```
> plotMDS(d)
```



The DCLK and WT samples separate quite nicely.

3.2.5 Estimating the dispersion

First we estimate the common dispersion to get an idea of the overall degree of inter-library variability in the data:

```
> d <- estimateCommonDisp(d, verbose=TRUE)
Disp = 0.151 , BCV = 0.389
```

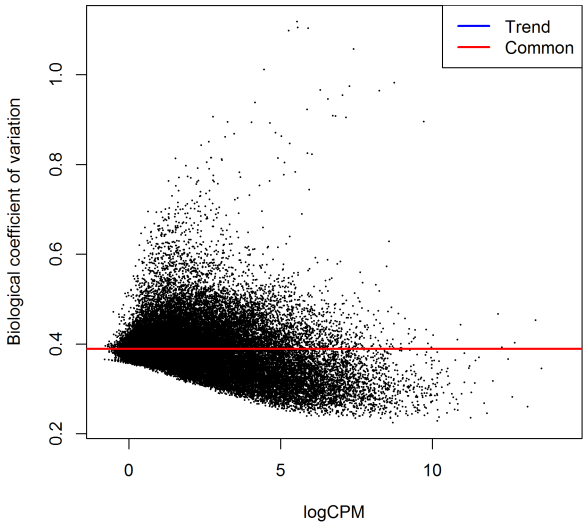
The biological coefficient of variation is the square root of the common dispersion.

Generally it is important to allow tag-specific dispersion estimates, so we go on to compute empirical Bayes moderated tagwise dispersion estimates. The trend is turned off as not usually required for SAGE data:

```
> d <- estimateTagwiseDisp(d, trend="none")
```

The following plot displays the estimates:

```
> plotBCV(d)
```



3.2.6 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 4.65e-20 2.09e-15
CATAAGTCACAGAGTCG  9.83   3.48 2.28e-18 5.12e-14
CCAAGAATCTGGTCGTA  3.91   3.54 7.67e-15 1.15e-10
GCTAATAAATGGCAGAT  3.19   5.82 5.50e-14 6.17e-10
CTGCTAAGCAGAAGCAA  3.42   3.80 7.45e-14 6.69e-10
AAAAGAAATCACAGTTG  9.49   3.07 1.57e-13 1.17e-09
TTCCTGAAAATGTGAAG  3.66   3.81 2.43e-13 1.56e-09
ATACTGACATTTCGTAT -4.32   4.32 3.50e-13 1.97e-09
CTACTGCAGCATTATCG  3.03   3.95 1.94e-12 9.68e-09
CTGACCCACTCAATGCT  3.50   3.98 6.41e-12 2.88e-08
```

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:

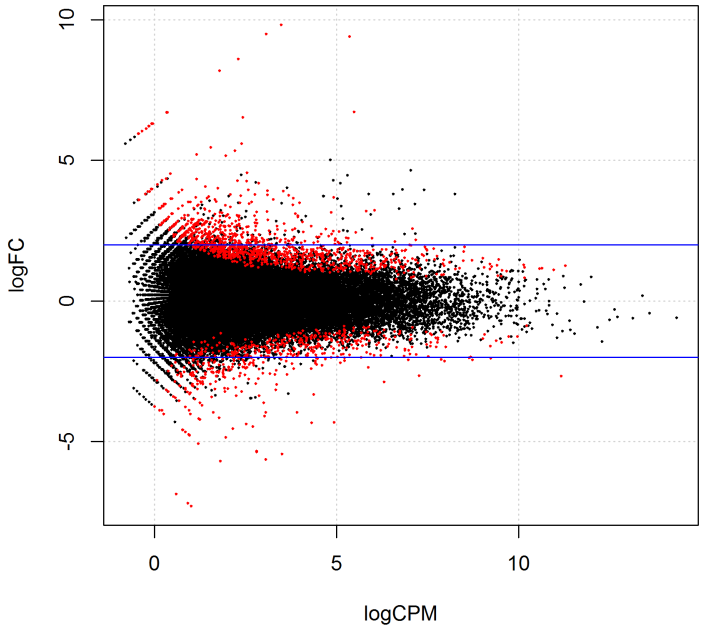
```
> detags <- rownames(topTags(et)$table)
> cpm(d)[detags, order(d$samples$group)]
                   DCLK1  DCLK2  DCLK3  DCLK4    WT1    WT2   WT3   WT4
TCTGTACGCAGTCAGGC  59.58  31.54 178.81  50.68  0.000  0.281  0.00  0.00
CATAAGTCACAGAGTCG  24.95  24.05  23.57  10.75  0.000  0.000  0.00  0.00
CCAAGAATCTGGTCGTA  26.07  20.61  19.10  19.96  0.853  1.405  0.00  2.23
GCTAATAAATGGCAGAT 144.11 100.24  53.64 109.03 12.791  8.993  3.39 12.09
CTGCTAAGCAGAAGCAA  28.30  27.48  21.13  23.04  1.990  1.967  0.00  3.50
AAAAGAAATCACAGTTG  11.54  28.11  17.07   4.61  0.000  0.000  0.00  0.00
TTCCTGAAAATGTGAAG  27.56  21.86  34.95  15.36  1.706  2.529  0.00  2.23
ATACTGACATTTCGTAT   1.86   1.56   3.25   1.54 32.121 64.076 13.56 33.10
CTACTGCAGCATTATCG  27.56  30.29  28.45  19.96  3.411  3.091  0.00  4.14
CTGACCCACTCAATGCT  23.09  23.73  45.92  19.96  3.980  1.686  0.00  2.23
```

The total number of differentiallly expressed genes at FDR$<$ 0.05:

```
> summary(de <- decideTestsDGE(et, p=0.05))
   [,1]
-1   540
0  43215
1   1127
```

A smearplot displays the log-fold changes with the DE genes highlighted:

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



Blue lines indicate 4-fold changes.

3.2.7 Setup

This analysis was conducted on:

```
> sessionInfo()
R version 2.15.0 (2012-03-30)
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_2.7.4  limma_3.13.1

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

3.3 RNA-seq of hormone-treated LNCaP cells

3.3.1 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.

3.3.2 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.

3.3.3 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.

3.3.4 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))
```

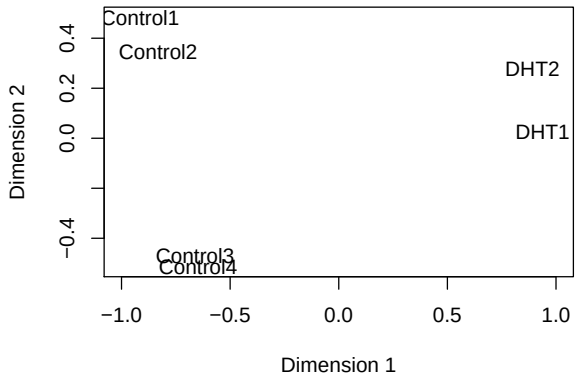
```
> d <- DGEList(counts = d, group = group)
> dim(d)
[1] 37435     7
> cpm.d <- cpm(d)
> d <- d[ rowSums(cpm.d > 1) >=3, ]
> d <- calcNormFactors(d)
```

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

3.3.5 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.

3.3.6 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, verbose=TRUE)
Disp = 0.02 , BCV = 0.141
```

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.

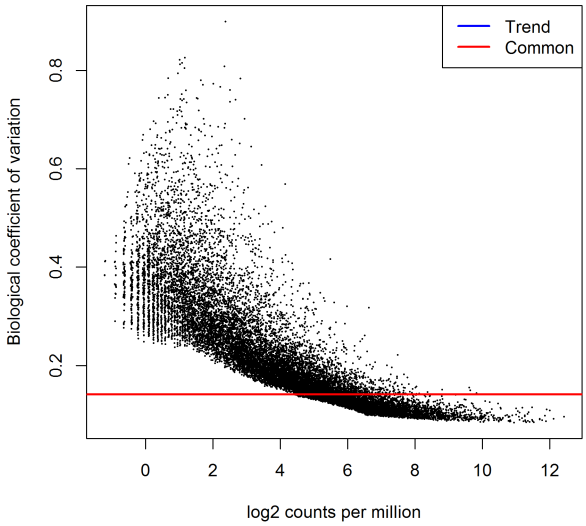
An extension to simply using the common dispersion for each tag is to estimate the dispersion separately for each tag, while 'squeezing' these estimates towards the common dispersion estimate in order to improve inference by sharing information between tags. This type of analysis can also be carried 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). 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 BCV estimates decrease as the tag abundance increases.

```
> plotBCV(d)
```



3.3.7 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
```

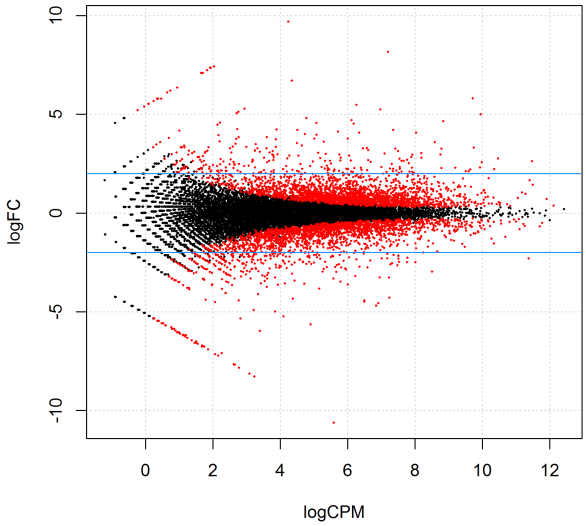
The `decideTestsDGE` function provides a useful way to summarize DE results after testing, as shown below.

```
> 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")
```



3.3.8 Setup

The analysis of this section was conducted with:

```
> sessionInfo()
R version 2.15.0 (2012-03-30)
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_2.7.3  limma_3.13.1

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

3.4 RNA-Seq of oral carcinomas vs matched normal tissue

3.4.1 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]. This provides an example of the GLM capabilities of edgeR.

RNA was sequenced on an Applied Biosystems SOLiD System 3.0 and reads mapped to the UCSC hg18 reference genome Tuch et al. [2010]. Read counts, summarised at the level of refSeq transcripts, are available in Table S1 of Tuch et al. [2010].

3.4.2 Reading in the data

The read counts for the six individual libraries are stored in one tab-delimited file. To make this file, we downloaded Table S1 from Tuch et al. [2010], deleted some unnecessary columns and edited the column headings slightly:

```
> rawdata <- read.delim("TableS1.txt", check.names=FALSE, stringsAsFactors=FALSE)
> head(rawdata)
      RefSeqID    Symbol NbrOfExons   8N 8T   33N 33T   51N  51T
1    NM_182502 TMPRSS11B         10 2592  3  7805 321  3372    9
2    NM_003280     TNNC1          6 1684  0  1787   7  4894  559
3    NM_152381     XIRP2         10 9915 15 10396  48 23309 7181
4    NM_022438       MAL          3 2496  2  3585 239  1596    7
5 NM_001100112      MYH2         40 4389  7  7944  16  9262 1818
6    NM_017534      MYH2         40 4402  7  7943  16  9244 1815
```

For easy manipulation, we put the data into a `DGEList` object:

```
> library(edgeR)
> y <- DGEList(counts=rawdata[,4:9], genes=rawdata[,1:3])
```

3.4.3 Annotation

The study by Tuch et al. [2010] was undertaken a few years ago, so not all of the RefSeq IDs provided by match RefSeq IDs currently in use. We retain only those transcripts with IDs in the current NCBI annotation, which is provided by the `org.HS.eg.db` package:

```
> library(org.Hs.eg.db)
> idfound <- y$genes$RefSeqID %in% mappedRkeys(org.Hs.egREFSEQ)
> y <- y[idfound,]
> dim(y)
[1] 15610     6
```

We add Entrez Gene IDs to the annotation:

```
> egREFSEQ <- toTable(org.Hs.egREFSEQ)
> head(egREFSEQ)
  gene_id accession
1       1 NM_130786
2       1 NP_570602
3       2 NM_000014
4       2 NP_000005
5       3 NR_040112
6       9 NM_000662
> m <- match(y$genes$RefSeqID, egREFSEQ$accession)
> y$genes$EntrezGene <- egREFSEQ$gene_id[m]
```

Now use the Entrez Gene IDS to update the gene symbols:

```
> egSYMBOL <- toTable(org.Hs.egSYMBOL)
> head(egSYMBOL)
  gene_id symbol
1       1   A1BG
2       2    A2M
3       3  A2MP1
4       9   NAT1
5      10   NAT2
6      11   AACP
> m <- match(y$genes$EntrezGene, egSYMBOL$gene_id)
> y$genes$Symbol <- egSYMBOL$symbol[m]
> head(y$genes)
      RefSeqID    Symbol NbrOfExons EntrezGene
1    NM_182502 TMPRSS11B         10     132724
2    NM_003280     TNNC1          6       7134
3    NM_152381     XIRP2         10     129446
4    NM_022438       MAL          3       4118
5 NM_001100112      MYH2         40       4620
6    NM_017534      MYH2         40       4620
```

3.4.4 Filtering

Different RefSeq transcripts for the same gene symbol count predominately the same reads. So we keep one transcript for each gene symbol. We choose the transcript with highest overall count:

```
> o <- order(rowSums(y$counts))
> y <- y[o,]
> d <- duplicated(y$genes$Symbol)
> y <- y[!d,]
> nrow(y)
```

```
[1] 10529
```

Normally we would also filter lowly expressed genes. For this data, all transcripts already have at least 50 reads for all samples of at least one of the tissues types.

Recompute the library sizes:

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

Use Entrez Gene IDs as row names:

```
> rownames(y$counts) <- rownames(y$genes) <- y$genes$EntrezGene
> y$genes$EntrezGene <- NULL
```

3.4.5 Normalization

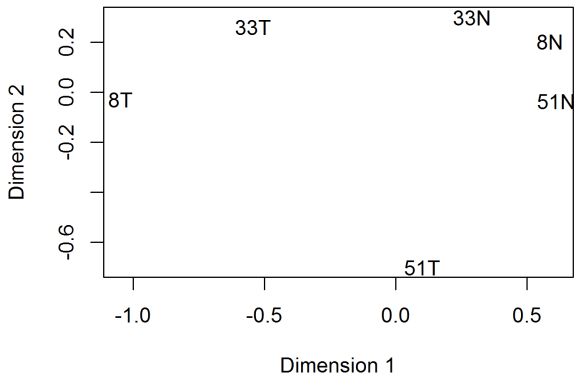
TMM normalization is applied to this dataset to account for compositional difference between the libraries.

```
> y <- calcNormFactors(y)
> y$samples
    group lib.size norm.factors
8N      1  7413954        1.155
8T      1  7140100        1.062
33N     1 15308019        0.656
33T     1 13704190        0.948
51N     1 19374726        1.089
51T     1 14430515        1.203
```

3.4.6 Data exploration

The first step of an analysis should be to examine the samples for outliers and for other relationships. The function `plotMDS` produces a plot in which distances between samples correspond to leading biological coefficient of variation (BCV) between those samples:

```
> plotMDS(y)
```



In the plot, dimension 1 separates the tumor from the normal samples, while dimsionion 2 roughly corresponds to patient number. This confirms the paired nature of the samples. The tumor samples appear more heterogeneous than the normal samples.

3.4.7 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:

```
> Patient <- factor(c(8,8,33,33,51,51))
> Tissue <- factor(c("N","T","N","T","N","T"))
> data.frame(Sample=colnames(y),Patient,Tissue)
  Sample Patient Tissue
1     8N       8      N
2     8T       8      T
3    33N      33      N
4    33T      33      T
5    51N      51      N
6    51T      51      T
> design <- model.matrix(~Patient+Tissue)
> rownames(design) <- colnames(y)
```

This sort of additive model is appropriate for paired designs, or experiments with batch effects.

3.4.8 Estimating the dispersion

First we estimate the overall dispersion for the dataset, to get an idea of the overall level of biological variability:

```
> y <- estimateGLMCommonDisp(y, design, verbose=TRUE)
Disp = 0.162 , BCV = 0.402
```

The square root of the common dispersion gives the coefficient of variation of biological variation. Here the common dispersion is found to be 0.162, so the coefficient of biological variation is around 0.402:

Then we estimate gene-wise dispersion estimates, allowing a possible trend with averge count size:

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

3.4.9 Differential expression

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

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

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

```
> lrt <- glmLRT(y, fit)
> topTags(lrt)
Coefficient:  TissueT
           RefSeqID    Symbol NbrOfExons logFC logCPM    LR   PValue      FDR
27179     NM_014440     IL36A          4 -6.13   5.48 108.6 2.00e-25 1.97e-21
4118      NM_022440       MAL          2 -7.16   6.66 107.3 3.74e-25 1.97e-21
5837      NM_005609      PYGM         20 -5.48   6.07  98.5 3.32e-23 1.17e-19
5737      NM_000959     PTGFR          3 -5.21   4.81  94.7 2.22e-22 5.86e-19
132724    NM_182502 TMPRSS11B         10 -7.42   7.72  86.9 1.14e-20 2.39e-17
487       NM_173201    ATP2A1         22 -4.62   6.03  83.9 5.11e-20 8.97e-17
3850      NM_057088      KRT3          9 -5.83   6.57  81.6 1.65e-19 2.48e-16
4606      NM_004533    MYBPC2         28 -5.47   6.57  80.7 2.61e-19 3.44e-16
2027      NM_053013      ENO3         12 -5.18   6.39  74.3 6.55e-18 7.66e-15
1160   NM_001099735     CKMT2         10 -5.50   4.77  73.2 1.15e-17 1.21e-14
```

The top DE tags have tiny p-values and FDR values, as well as large fold changes.

Here `glmLFT` has conducted a test for the last coefficient in the linear model, which we can see is the tumor vs normal tissue effect:

```
> colnames(design)
[1] "(Intercept)" "Patient33"   "Patient51"   "TissueT"
```

The genewise tests for tumor vs normal differential expression, adjusting for differences between the three patients, can be viewed as analogous to a paired t-tests, but generalized to negative binomial count data.

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

```
> o <- order(lrt$table$PValue)
> cpm(y)[o[1:10],]
          8N   8T   33N    33T   51N     51T
27179   49.5 1.40 119.2  3.284  41.4  0.0693
4118   279.6 0.14 191.9  7.005  69.7  0.4851
5837   188.7 3.08  82.8  1.168 112.1  7.1377
5737    61.4 0.98  18.5  0.876  89.5  3.1184
132724 349.6 0.42 509.9 23.423 174.0  0.6237
487    131.9 3.50 101.4  3.794 116.9 11.1569
3850   144.2 0.98 246.5 26.123  45.7  0.3465
4606   130.3 1.54  31.7  0.438 415.2 31.6690
2027   146.3 0.56  84.1  5.400 249.4 15.4534
1160    48.2 0.14  23.9  0.511  85.9  6.7219
```

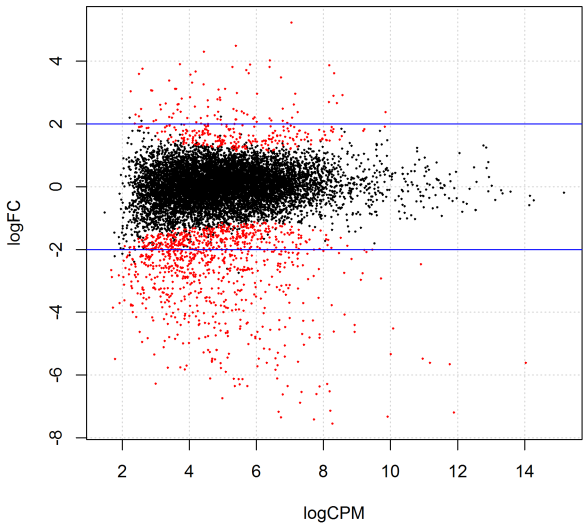
We see that all the top genes have consistent tumour vs normal changes for the three patients.

The total number of differentially expressed genes at 5% FDR is given by:

```
> summary(de <- decideTestsDGE(lrt))
   [,1]
-1  978
0  9247
1   304
```

Plot log-fold change against log-counts per million, with DE genes highlighted:

```
> detags <- rownames(y)[as.logical(de)]
> plotSmear(lrt, de.tags=detags)
> abline(h=c(-2, 2), col="blue")
```



The blue lines indicate 4-fold changes.

3.4.10 Setup

This analysis was conducted on:

```
> sessionInfo()
R version 2.15.0 (2012-03-30)
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_2.7.1   RSQLite_0.11.1       DBI_0.2-5
[4] AnnotationDbi_1.18.0 Biobase_2.16.0       BiocGenerics_0.2.0
[7] edgeR_2.7.8          limma_3.13.1

loaded via a namespace (and not attached):
[1] IRanges_1.14.2 stats4_2.15.0  tools_2.15.0
```

3.5 RNA-Seq of pathogen inoculated arabidopsis with batch effects

3.5.1 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.

3.5.2 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 *P. 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.

3.5.3 Sequencing

The six RNA samples were sequenced one per lane on 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`).

3.5.4 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 that each replicate was conducted:

```
> 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 two counts 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 subsequent 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
```

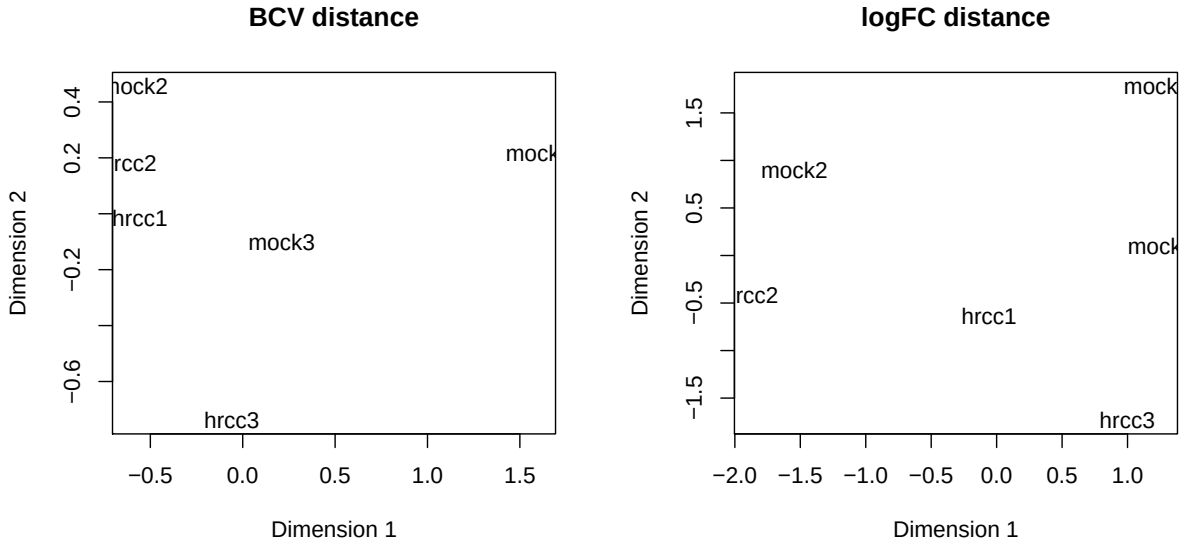
3.5.5 Data exploration

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.

```
> design <- model.matrix(~Time+Time:Treat)
> logFC <- predFC(y,design,prior.count=1)/log(2)
```

The logFC at the three times are positively correlated with one another, as we would hope:

```
> 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.

3.5.6 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"
```

3.5.7 Estimating the dispersion

Estimate the average dispersion over all genes:

```
> y <- estimateGLMCommonDisp(y, design, verbose=TRUE)
Disp = 0.0706 , BCV = 0.266
```

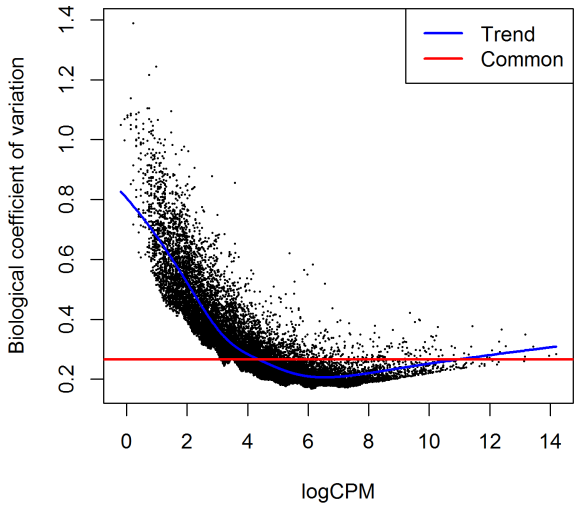
The square root of dispersion is the coefficient of biological variation (BCV). Here the common dispersion is 0.0706, so the BCV is 0.266. The common BCV is on the high side, considering that this is a designed experiment using genetically identical plants.

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

```
> y <- estimateGLMTrendedDisp(y, design)
> y <- estimateGLMTagwiseDisp(y, design, prior.n=3)
```

Here we have chosen `prior.n` slightly smaller than the default, which is 10 in this case, after inspecting the following BCV plot. The genewise dispersions show a decreasing trend with expression level. At low logCPM, the dispersions are very large indeed:

```
> plotBCV(y)
```



3.5.8 Differential expression

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

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

First we check whether there was a genuine need to adjust for the experimental times. We do this by testing for differential expression between the three times. There is considerable differential expression, justifying our decision to adjust for the batch effect:

```
> lrt <- glmLRT(y, fit, coef=2:3)
> topTags(lrt)
Coefficient:  Time2 Time3
          logFC.Time2 logFC.Time3 logCPM  LR   PValue      FDR
AT5G66800        5.59      -1.075   5.43 271 1.61e-59 2.66e-55
AT5G31702        5.84      -2.605   5.90 223 3.01e-49 2.49e-45
AT5G23000        5.62      -0.289   5.68 199 6.28e-44 3.46e-40
AT3G33004        4.82      -1.764   5.60 195 4.38e-43 1.81e-39
AT2G45830        5.43      -0.596   4.65 181 4.01e-40 1.33e-36
AT2G11230        3.50      -1.532   5.56 166 8.64e-37 2.38e-33
AT2G07782        3.49      -1.618   5.23 151 1.59e-33 3.75e-30
AT2G23910        3.60      -0.386   5.07 141 1.95e-31 4.03e-28
AT5G35736        5.44      -0.994   4.57 134 7.95e-30 1.42e-26
AT2G27770        2.47      -1.571   5.37 134 8.60e-30 1.42e-26
> FDR <- p.adjust(lrt$table$PValue, method="BH")
> sum(FDR < 0.05)
[1] 3276
```

Now conduct likelihood ratio tests for the pathogen effect and show the top genes. By default, the test is for the last coefficient in the design matrix, which in this case is the treatment effect:

```
> lrt <- glmLRT(y, fit)
> topTags(lrt)
Coefficient:  Treathrcc
          logFC logCPM  LR   PValue      FDR
AT2G19190  4.50   7.37 255 2.38e-57 3.93e-53
AT5G48430  6.34   6.71 231 2.86e-52 2.36e-48
AT2G39530  4.34   6.70 220 9.01e-50 4.96e-46
AT2G39380  4.95   5.75 201 1.01e-45 4.19e-42
AT3G46280  4.78   8.09 194 4.76e-44 1.57e-40
AT1G51800  3.97   7.70 192 9.60e-44 2.64e-40
AT2G44370  5.43   5.17 174 1.20e-39 2.83e-36
AT1G51850  5.33   5.39 167 2.80e-38 5.79e-35
AT1G51820  4.34   6.36 162 5.07e-37 9.31e-34
AT3G55150  5.80   4.86 158 2.74e-36 4.53e-33
```

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
AT2G19190 358.6 279.1 327.3 16.343 12.64 12.00
AT5G48430 198.6 344.7 116.6  4.218  4.74  0.00
AT2G39530 166.1 210.3 226.7  6.854  9.48 12.00
AT2G39380  96.3  92.5 126.0  2.109  3.16  4.31
AT3G46280 404.4 410.3 765.3 18.452 17.91 16.62
AT1G51800 380.8 381.0 432.6 28.469 17.38 27.70
AT2G44370  59.9  73.5  80.2  2.109  1.05  1.54
AT1G51850  82.1  61.7 101.5  1.054  1.05  3.39
AT1G51820 127.4 171.6 178.3  9.490  7.90  5.54
AT3G55150  45.3  71.2  60.0  0.527  1.05  1.23
```

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

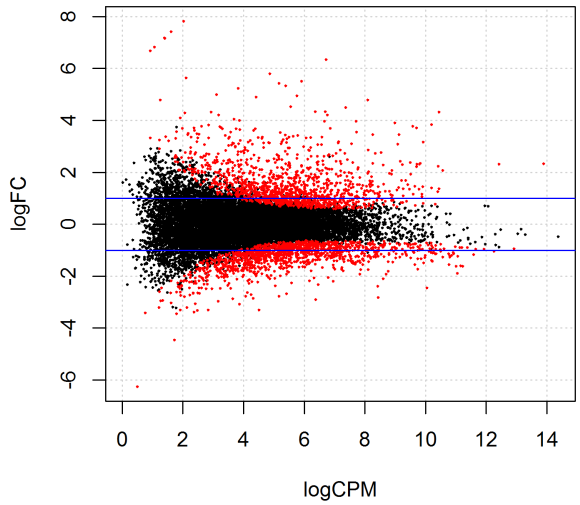
```
> summary(dt <- decideTestsDGE(lrt))
   [,1]
-1  1291
0  13959
1   1276
```

We can pick out which genes are DE:

```
> isDE <- as.logical(dt)
> DEnames <- rownames(y)[isDE]
```

Then we can plot all the logFCs against average count size, highlighting the DE genes:

```
> plotSmear(lrt, de.tags=DEnames)
> abline(h=c(-1,1), col="blue")
```



The blue lines indicate 2-fold up or down.

3.5.9 Setup

This analysis was conducted on:

```
> sessionInfo()
R version 2.15.0 (2012-03-30)
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_2.7.8   limma_3.13.1  NBPSeq_0.1.4  qvalue_1.30.0

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

Bibliography

Simon Anders and Wolfgang Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11(10):R106, Oct 2010. doi: 10.1186/gb-2010-11-10-r106.

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.

JH Bullard, E Purdom, KD Hansen, and S Dudoit. Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments. *BMC Bioinformatics*, 18:11–94, February 2010.

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.

J Lu, JK Tomfohr, and TB Kepler. Identifying differential expression in multiple SAGE libraries: an overdispersed log-linear model approach. *BMC Bioinformatics*, 6:165, 2005. URL `http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=pubmed&cmd=Retrieve&dopt=AbstractPlus&list_uids=17713979177590352770related:gmvwljWy1PUJ`.

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`.

P. McCullagh and John A. Nelder. *Generalized Linear Models*. Chapman & Hall/CRC, Boca Raton, Florida, 2nd edition edition, 1989.

J. A Nelder and R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3):370–384, 1972. URL `http://www.jstor.org/stable/2344614`.

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.