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 14 December 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$^{\text{TM}}$, 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 when you publish results obtained using the software, 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 is a crucial improvement on the previous idea of estimating the dispersions from a global model, because it permits gene-specific dispersion estimation to be reliable even for small samples. Gene-specific dispersion estimation is necessary so that genes that behave consistently across replicates should rank more highly than genes that do not.

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* 40, 4288-4297.

> 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 more complete explanation of the concept of *biological coefficient of variation*.

Lund, SP, Nettleton, D, McCarthy, DJ, Smyth, GK (2012). Detecting differential expression in RNA-sequence data using quasi-likelihood with shrunken dispersion estimates. *Statistical Applications in Genetics and Molecular Biology.* (Accepted 31 July 2012)

This paper explains the `glmQLFTest` function, which is an alternative to `glmLRT`, and which replaces the chisquare approximation to the likelihood ratio statistic with a quasi-likelihood F-test.

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. All other questions about how to use edgeR are best sent to the Bioconductor mailing list `bioconductor@stat.math.ethz.ch`. 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`.

Posting questions to the Bioconductor mailing list has a number of advantages. First, the mailing list includes a community of experienced edgeR users who can answer most common questions. Second, the edgeR authors try hard to ensure that any user posting to Bioconductor receives assistance. Third, the mailing list allows others with the same sort of questions to gain from the answers. The authors do occasionally answer questions posted to other forums such as SEQAnswers, but it is not possible to do this on regular basis. We recommend therefore that questions be posted to Bioconductor.

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 <- calcNormFactors(y)
> 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 <- estimateGLMCommonDisp(y,design)
> y <- estimateGLMTrendedDisp(y,design)
> y <- estimateGLMTagwiseDisp(y,design)
> fit <- glmFit(y,design)
> lrt <- glmLRT(fit,coef=2)
> topTags(lrt)
```

Many variants are available on this analysis.

Chapter 2

Overview of capabilities

2.1 Terminology

edgeR performs differential abundance analysis for pre-defined genomic features. Although not strictly necessary, it usually desirable that these genomic features are non-overlapping. For simplicity, we will hence-forth refer to the genomic features as "genes", although they could in principle be transcripts, exons, general genomic intervals or some other type of feature. For ChIP-seq experiments, abundance might relate to transcription factor binding or to histone mark occupancy, but we will henceforth refer to abundance as in terms of gene expression. In other words, the remainder of this guide will use terminology as for a gene-level analysis of an RNA-seq experiment, although the methodology is more widely applicable than that.

2.2 Producing a table of counts

edgeR works on a table of integer read counts, with rows corresponding to genes and columns to independent libraries. The counts represent the total number of reads aligning to each gene (or other genomic locus). Such counts can be produced from aligned reads by a variety of short read software tools, for example `featureCounts` in the Rsubread package, `findOverlaps` in the GenomicRanges package, or the Python software `htseq-counts`.

When conducting gene-level analyses, the counts could be for reads mapping anywhere in the genomic span of the gene or the counts could be for exons only. For routine use, we generally recommend counting reads mapping to exons including the UTRs.

2.3 Reading the counts from a file

If the table of counts has been written to a file, then the first step in any analysis will usually be to read these counts into an R session.

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.

2.4 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.

2.5 Normalization

2.5.1 Normalization is only necessary for sample-specific effects

edgeR is concerned with differential expression analysis rather than with the quantification of expression levels. It is concerned with relative changes in expression levels between conditions, but not directly with estimating absolute expression levels. This greatly simplifies the technical influences that need to be taken into account, because any technical factor that is unrelated to the experimental conditions should cancel out of any differential expression analysis. For example, read counts can generally be expected to be proportional to length as well as to expression for any transcript, but edgeR does not generally need to adjust for gene length because gene length has the same relative influence on the read counts for each RNA sample. For this reason, normalization issues arise only to the extent that technical factors have sample-specific effects.

2.5.2 Sequencing depth

The most obvious technical factor that affects the read counts, other than gene expression levels, is the sequencing depth of each RNA sample. edgeR adjusts any differential expression analysis for varying sequencing depths as represented by differing library sizes. This is part of the basic modeling procedure and flows automatically into fold-change or p-value calculations. It is always present, and doesn't require any user intervention.

2.5.3 RNA composition

The second most important technical influence on differential expression is one that is less obvious. RNA-seq provides a measure of the relative abundance of each gene in each RNA sample, but does not provide any measure of the total RNA output on a per-cell basis. This commonly becomes important when a small number of genes are very highly expressed in one sample, but not in another. The highly expressed genes can consume a substantial proportion of the total library size, causing the remaining genes to be under-sampled in that sample. Unless this *RNA composition* effect is adjusted for, the remaining genes may falsely appear to be down-regulated in that sample [Robinson and Oshlack, 2010].

The `calcNormFactors` function normalizes for RNA composition by finding a set of scaling factors for the library sizes that minimize the log-fold changes between the samples for most genes. The default method for computing these scale factors uses a trimmed mean of M-values (TMM) between each pair of samples [Robinson and Oshlack, 2010]. We call the product of the original library size and the scaling factor the *effective library size*. The effective library size replaces the original library size in all downsteam analyses.

2.5.4 GC content

The GC-content of each gene does not change from sample to sample, so it can be expected to have little effect on differential expression analyses to a first approximation. Recent publications, however, have demonstrated that sample-specific effects for GC-content can be detected [Risso et al., 2011, Hansen et al., 2012]. The EDASeq [Risso et al., 2011] and cqn [Hansen et al., 2012] packages estimate correction factors that adjust for sample-specific GC-content effects in a way that is compatible with edgeR. In each case, the observation-specific correction factors can be input into the glm functions of edgeR as an *offset* matrix.

2.5.5 Gene length

Like GC-content, gene length does not change from sample to sample, so it can be expected to have little effect on differential expression analyses. Nevertheless, sample-specific effects for gene length have detected [Hansen et al., 2012], although the evidence is not as strong as for GC-content.

2.5.6 Model-based normalization, not transformation

In edgeR, normalization takes the form of correction factors that enter into the statistical model. Such correction factors are usually computed internally by edgeR functions, but it is also possible for a user to supply them. The correction factors may take the form of scaling factors for the library sizes, such as computed by `calcNormFactors`, which are then used to compute the effective library sizes. Alternatively, gene-specific correction factors can be entered into the glm functions of edgeR as offsets. In the latter case, the offset matrix will be assumed to account for all normalization issues, including sequencing depth and RNA composition.

Note that normalization in edgeR is model-based, and the original read counts are not themselves transformed. This means that users should not transform the read counts in any way before inputing them to edgeR. For example, users should not enter RPKM or FPKM values to edgeR in place of read counts. Such quantities will prevent edgeR from correctly estimating the mean-variance relationship in the data, which is a crucial to the statistical strategies underlying edgeR. Similarly, edgeR is not designed to work with estimated expression levels, for example as might be output by Cufflinks.

2.6 Negative binomial models

2.6.1 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 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.

2.6.2 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:

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

Dividing both sides by μ_{gi}^2 gives

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

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.

2.6.3 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 (or the dispersion trend) and the precision of the tagwise estimates, and can be considered as the prior degrees of freedom. The default behavior of the edgeR is to set the prior degrees of freedom to 20 based on the past experience with a number of data sets, although some smaller values are suitable for some particular RNA-Seq data sets.

2.7 Pairwise comparisons between two or more groups (classic)

2.7.1 Estimating dispersions

edgeR uses the quantile-adjusted conditional maximum likelihood (qCML) method for 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 numbers of replicates for SAGE and RNA-seq experiments.

The qCML method calculates the likelihood by conditioning on the total counts for each tag, and uses pseudo counts after adjusting for library sizes. Given a table of 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 datasets with a 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 a single factor. When an experiment has more than one factor involved, we need to seek a new way of estimating dispersions.

Here is a simple example of 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 4.1 (Zhang's data), section 4.2 ('t Hoen's data) and section 4.3 (Li's data).

2.7.2 Testing for DE genes

For all the Next-Gen squencing data analyses we consider here, people are most interested in finding differentially expressed genes/tags between two (or more) groups. Once negative binomial models are fitted and dispersion estimates are obtained, we can proceed with testing procedures for determining 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 4.1 (Zhang's data), section 4.2 ('t Hoen's data) and section 4.3 (Li's data).

2.8 More complex experiments (glm functionality)

2.8.1 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.

2.8.2 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 the tagwise dispersion approach is strongly recommended in multi-factor experiment cases.

Here is a simple example of estimating dispersions using the 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 4.4 (Tuch's data).

2.8.3 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 experiments 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.

This DGEGLM object can then be passed to the function glmLRT() to carry out the likelihood ratio test. User can select one or more coefficients to drop from the full design matrix. This gives the null model against which the full model is compared using 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(fit,coef=2)
> topTags(lrt.2vs1)
```

To compare 3 vs 1:

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

To compare 3 vs 2:

```
> lrt.3vs2 <- glmLRT(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(fit,coef=2:3)
> topTags(lrt)
```

For more detailed examples, see the case study in section 4.4 (Tuch's data) and 4.5 (arabidopsis RNA-Seq data).

2.9 What to do if you have no replicates

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

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

2. Simply pick a reasonable dispersion value, based on your experience with similar data, and use that for `exactTest` or `glmFit`. Typical values for the common BCV (square-root-dispersion) for datasets arising from well-controlled experiments are 0.4 for human data, 0.1 for data on genetically identical model organisms or 0.01 for technical replicates. Here is a toy example with simulated data:

   ```
   > bcv <- 0.2
   > counts <- matrix( rnbinom(40,size=1/bcv^2,mu=10), 20,2)
   > y <- DGEList(counts=counts, group=1:2)
   > et <- exactTest(y, dispersion=bcv^2)
   ```

   Note that the p-values obtained and the number of significant genes will be very sensitive to the dispersion value chosen, and be aware than less well controlled datasets, with unaccounted-for batch effects and so on, could have in reality much larger dispersions than are suggested here. Nevertheless, choosing a nominal dispersion value may be more realistic than ignoring biological variation entirely.

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, then 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.

2.10 Clustering, heatmaps etc

edgeR provides the function `plotMDS` to draw a multi-dimensional scaling plot of the RNA samples in which distances correspond to BCV between each pair of the samples for the most heterogeneous genes. This plot can be viewed as a type of unsupervised clustering.

Inputing RNA-seq counts to clustering or heatmap routines designed for microarray data is not straight-forward, and the best way to do this is still a matter of research. To draw a heatmap of individual RNA-seq samples, we suggest using output from `predFC`, for example

```
> y <- predFC(d, prior.count.total=2*ncol(d))
```

where `d` is the normalized `DGEList` object. This produces a matrix of $\log_2$ counts-per-million, with undefined values avoided and the poorly defined log-fold-changes for low counts shrunk towards zero. Larger values for `prior.count.total` produce more shrinkage. The logCPM values could optionally be converted to RPKM or FPKM by subtracting $\log_2$ of gene length. The Arabidopsis case study of Section 4.5 gives two examples of this in conjunction with MDS plots, one example making a plot from the log-counts-per-million and another making a plot of shrunk log-fold-changes.

Chapter 3

Specific Experimental Designs

3.1 Introduction

In this chapter, we outline the principles for setting up the design matrix and forming contrasts for some typical experimental designs.

3.2 Two or more Groups

3.2.1 Introduction

The simplest and most common type of experimental design is that in which a number of experimental conditions are compared on the basis of independent biological replicates of each condition. Suppose that there are three experimental conditions to be compared, treatments A, B and C, say. The `samples` component of the `DGEList` data object might look like:

```
> y$samples
         group lib.size norm.factors
sample.1     A   100001            1
sample.2     A   100002            1
sample.3     B   100003            1
sample.4     B   100004            1
sample.5     C   100005            1
```

Note that it is not necessary to have multiple replicates for all the conditions, although it is usually desirable to do so. By default, the conditions will be listed in alphabetical order, regardless of the order that the data were read:

```
> levels(y$samples$group)
[1] "A" "B" "C"
```

3.2.2 Classic approach

The classic edgeR approach is to make pairwise comparisons between the groups. For example,

```
> et <- exactTest(y, pair=c("A","B"))
> topTags(et)
```

will find genes differentially expressed (DE) in B vs A. Similarly

```
> et <- exactTest(y, pair=c("A","C"))
```

for C vs A, or

```
> et <- exactTest(y, pair=c("C","B"))
```

for B vs C.

Alternatively, the conditions to be compared can be specified by number, so that

```
> et <- exactTest(y, pair=c(3,2))
```

is equivalent to `pair=c("C","B")`, given that the second and third levels of `group` are `B` and `C` respectively.

Note that the levels of `group` are in alphabetical order by default, by can be easily changed. Suppose for example that C is a control or reference level to which conditions A and B are to be compared. Then one might redefine the group levels, in a new data object, so that C is the first level:

```
> y2 <- y
> y2$samples$group <- relevel(y2$samples$group, ref="C")
> levels(y2$samples$group)
[1] "C" "A" "B"
```

Now

```
> et <- exactTest(y2, pair=c("A","B"))
```

would still compare B to A, but

```
> et <- exactTest(y2, pair=c(1,2))
```

would now compare A to C.

When `pair` is not specified, the default is to compare the first two group levels, so

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

compares B to A, whereas

```
> et <- exactTest(y2)
```

compares A to C.

3.2.3 GLM approach

The glm approach to multiple groups is similar to the classic approach, but permits more general comparisons to be made. The glm approach requires a design matrix to describe the treatment conditions. We will usually use the `model.matrix` function to construct the design matrix, although it could be constructed manually. There are always many equivalent ways to define this matrix. Perhaps the simplest way is to define a coefficient for the expression level of each group:

```
> design <- model.matrix(~0+group, data=y$samples)
> colnames(design) <- levels(y$samples$group)
> design
         A B C
sample.1 1 0 0
sample.2 1 0 0
sample.3 0 1 0
sample.4 0 1 0
sample.5 0 0 1
```

Here, the `0+` in the model formula is an instruction not to include an intercept column and instead to include a column for each group.

One can compare any of the treatment groups using the `contrast` argument of the `glmLRT` function. For example,

```
> fit <- glmFit(y, design)
> lrt <- glmLRT(fit, contrast=c(-1,1,0))
> topTags(lrt)
```

will compare B to A. The meaning of the contrast is to make the comparison `-1*A + 1*B + 0*C`, which is of course is simply `B-A`.

The contrast vector can be constructed using `makeContrasts` if that is convenient. The above comparison could have been made by

```
> BvsA <- makeContrasts(B-A, levels=design)
> lrt <- glmLRT(fit, contrast=BvsA)
```

One could make three pairwise comparisons between the groups by

```
> my.contrasts <- makeContrasts(BvsA=B-A, CvsB=C-B, CvsA=A-C, levels=design)
> lrt.BvsA <- glmLRT(fit, my.contrasts[,"BvsA"])
> topTags(lrt.BvsA)
> lrt.CvsB <- glmLRT(fit, my.contrasts[,"CvsB"])
> topTags(lrt.CvsB)
> lrt.CvsA <- glmLRT(fit, my.contrasts[,"CvsA"])
> topTags(lrt.CvsA)
```

which would compare B to A, C to B and C to A respectively.

Any comparison can be made. For example,

```
> lrt <- glmLRT(fit, contrast=c(-0.5,-0.5,1))
```

would compare C to the average of A and B. Alternatively, this same contrast could have been specified by

```
> my.contrast <- makeContrasts(C-(A+B)/2, levels=design)
> lrt <- glmLRT(fit, contrast=my.contrast)
```

with the same results.

3.2.4 A more traditional glm approach

A more traditional way to create a design matrix in R is to include an intercept term that represents the first level of the factor. We included `0+` in our model formula above. Had we omitted it, the design matrix would have had the same number of columns as above, but the first column would be the intercept term and the meanings of the second and third columns would change:

```
> design <- model.matrix(~group, data=y$samples)
> design
         (Intercept) groupB groupC
sample.1           1      0      0
sample.2           1      0      0
sample.3           1      1      0
sample.4           1      1      0
sample.5           1      0      1
```

Now the first coefficient will measure the baseline logCPM expression level in the first treatment condition (here group A), and the second and third columns are relative to the baseline. Here the second and third coefficients represent B vs A and C vs A respectively. In other words, `coef=2` now means `B-A` and `coef=3` means `C-A`.

This parametrization makes good sense when one of the groups represents a reference or control group:

```
> design2 <- model.matrix(~group, data=y2$samples)
> design2
         (Intercept) groupA groupB
sample.1           1      1      0
sample.2           1      1      0
sample.3           1      0      1
sample.4           1      0      1
sample.5           1      0      0
```

Now

```
> fit2 <- glmFit(y2, design2)
> lrt <- glmLRT(fit2, coef=2)
```

compares A to C, and

```
> lrt <- glmLRT(fit2, coef=3)
```

compares B to C. With this parametrization, one could still compare B to A using

```
> lrt <- glmLRT(fit2, contrast=c(0,-1,1))
```

Note that

```
> lrt <- glmLRT(fit2, coef=1)
```

should not be used. It would test whether the first coefficient is zero, but it is not meaningful to compare the logCPM in group A to zero.

3.2.5 An ANOVA-like test for any differences

It might be of interest to find genes that are DE between any of the groups, without specifying before-hand which groups might be different. This is analogous to a one-way ANOVA test. In edgeR, this is done by specifying multiple coefficients to `glmLRT`, when the design matrix includes an intercept term. For example, with `fit` as above,

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

will find any genes that differ between any of the treatment conditions A, B or C. Technically, this procedure tests whether either of the contrasts `B-A` or `C-A` are non-zero. Since at least one of these must be non-zero when differences exist, the test will detect any differences. To have this effect, the `coef` argument should specify all the coefficients except the intercept.

Note that this approach does not depend on how the group factor was defined, or how the design matrix was formed, as long as there is an intercept column. For example

```
> lrt <- glmLRT(fit2, coef=2:3)
```

gives exactly the results, even though `fit2` and `fit` were computed using different design matrices.

3.3 Experiments with all combinations of multiple factors

3.3.1 Defining each treatment combination as a group

We now consider experiments with more than one experimental factor, but in which every combination of experiment conditions can potentially have a unique effect. For example, suppose that an experiment has been conducted with an active drug and a placebo, at three times from 0 hours to 2 hours, with all samples obtained from independent subjects. The data frame `targets` describes the treatment conditions applied to each sample:

```
> targets
    Sample   Treat Time
1  Sample1 Placebo   0h
2  Sample2 Placebo   0h
3  Sample3 Placebo   1h
4  Sample4 Placebo   1h
5  Sample5 Placebo   2h
6  Sample6 Placebo   2h
7  Sample1    Drug   0h
8  Sample2    Drug   0h
9  Sample3    Drug   1h
10 Sample4    Drug   1h
11 Sample5    Drug   2h
12 Sample6    Drug   2h
```

As always, there are many ways to setup a design matrix. A simple, multi-purpose approach is to combine all the experimental factors into one combined factor:

```
> Group <- factor(paste(targets$Treat,targets$Time,sep="."))
> cbind(targets,Group=Group)
```

Then we can take the same approach as in the previous section on two or more groups. Each treatment time for each treatment drug is a group:

```
> design <- model.matrix(~0+Group)
> colnames(design) <- levels(Group)
> fit <- glmFit(y, design)
```

Then we can make any comparisons we wish. For example, we might wish to make the following contrasts:

```
> my.contrasts <- makeContrasts(
+         Drug.1vs0 = Drug.1h-Drug.0h,
+         Drug.2vs0 = Drug.2h-Drug.0h,
+         Placebo.1vs0 = Placebo.1h-Placebo.0h,
+         Placebo.2vs0 = Placebo.2h-Placebo.0h,
+         DrugvsPlacebo.0h = Drug.0h-Placebo.0h,
+         DrugvsPlacebo.1h = (Drug.1h-Drug.0h)-(Placebo.1h-Placebo.0h),
+         DrugvsPlacebo.2h = (Drug.2h-Drug.0h)-(Placebo.2h-Placebo.0h),
+ levels=design)
```

To find genes responding to the drug at 1 hour:

```
> lrt <- glmLRT(fit, contrast=my.contrasts[,"Drug.1vs0"])
```

or at 2 hours:

```
> lrt <- glmLRT(fit, contrast=my.contrasts[,"Drug.2vs0"])
```

To find genes with baseline differences between the drug and the placebo at 0 hours:

```
> lrt <- glmLRT(fit, contrast=my.contrasts[,"DrugvsPlacebo.0h"])
```

To find genes that have responded *differently* to the drug at the placebo at 2 hours:

```
> lrt <- glmLRT(fit, contrast=my.contrasts[,"DrugvsPlacebo.2h"])
```

Of course, it is not compulsory to use `makeContrasts` to form the contrasts. The coefficients are the following:

```
> colnames(fit)
[1] "Drug.0h"    "Drug.1h"    "Drug.2h"    "Placebo.0h" "Placebo.1h" "Placebo.2h"
```

so

```
> lrt <- glmLRT(fit, contrast=c(-1,0,1,0,0,0))
```

would find the `Drug.2vs0` contrast, and

```
> lrt <- glmLRT(fit, contrast=c(-1,0,1,1,0,-1))
```

is another way of specifying the `DrugvsPlacebo.2h` contrast.

3.3.2 Nested interaction formulas

We generally recommend the approach of the previous section, because it is so explicit and easy to understand. However it may be useful to be aware of more short-hand approach to form the same contrasts in the previous section using a model formula. First, make sure that the placebo is the reference level:

```
> targets$Treat <- relevel(targets$Treat, ref="Placebo")
```

Then form the design matrix:

```
> design <- model.matrix(~Treat + Treat:Time, data=targets)
> fit <- glmFit(y, design)
```

The meaning of this formula is to consider all the levels of time for each treatment drug separately. The second term is a nested interaction, the interaction of Time within Treat. The coefficient names are:

```
> colnames(fit)
[1] "(Intercept)"         "TreatDrug"
[3] "TreatPlacebo:Time1h" "TreatDrug:Time1h"
[5] "TreatPlacebo:Time2h" "TreatDrug:Time2h"
```

Now most of the above contrasts are directly available as coefficients:

```
> lrt <- glmLRT(fit, coef=2)
```

is the baseline drug vs placebo comparison,

```
> lrt <- glmLRT(fit, coef=4)
```

is the drug effect at 1 hour,

```
> lrt <- glmLRT(fit, coef=6)
```

is the drug effect at 2 hours, and finally

```
> lrt <- glmLRT(fit, contrast=c(0,0,0,0-1,1))
```

is the `DrugvsPlacebo.2h` contrast.

3.3.3 Treatment effects over all times

The nested interaction model makes it easy to find genes that respond to the treatment at any time, in a single test. Continuing the above example,

```
> lrt <- glmLRT(fit, coef=c(4,6))
```

finds genes that respond to the treatment at either 1 hour or 2 hours versus the 0 hour baseline.

3.3.4 Interaction at any time

The full interaction formula is

```
> design <- model.matrix(~Treat * Time, data=targets)
```

which is equivalent to

```
> design <- model.matrix(~Treat + Time + Treat:Time, data=targets)
> fit <- glmFit(y, design)
```

This formula is primarily useful as a way to conduct an overall test for interaction. The coefficients are

The coefficient names are:

```
> colnames(design)
[1] "(Intercept)"      "TreatDrug"
[3] "Time1h"           "Time2h"
[5] "TreatDrug:Time1h" "TreatDrug:Time2h"
```

Now

```
> lrt <- glmLRT(fit, coef=2)
```

is again the baseline drug vs placebo comparison, but

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

and

```
> lrt <- glmLRT(fit, coef=4)
```

are the effects of the reference drug, i.e., the effects of the placebo at 1 hour and 2 hours. The last two coefficients give the `DrugvsPlacebo.1h` and `DrugvsPlacebo.2h` contrasts, so that

```
> lrt <- glmLRT(fit, coef=5:6)
```

is useful because it detects genes that respond *differently* to the drug, relative to the placebo, at either of the times.

3.4 Additive Models and Blocking

3.4.1 Paired Samples

Paired samples occur whenever we compare two treatments and each independent subject in the experiment receives both treatments. Suppose for example that an experiment is conducted to compare a new treatment (T) with a control (C). Suppose that both the control and the treatment are administered to each of three patients. This produces the sample data:

FileName	Subject	Treatment
File1	1	C
File2	1	T
File3	2	C
File4	2	T
File5	3	C
File6	3	T

This is a paired design in which each subject receives both the control and the active treatment. We can therefore compare the treatment to the control for each patient separately, so that baseline differences between the patients are subtracted out.

The design matrix is formed from an additive model formula without an interaction term:

```
> Subject <- factor(targets$Subject)
> Treat <- factor(targets$Treatment, levels=c("C","T"))
> design <- model.matrix(~Subject+Treat)
```

The omission of an interaction term is characteristic of paired designs. We are not interested in the effect of the treatment on an individual patient (which is what an interaction term would examine). Rather we are interested in the average effect of the treatment over a population of patients.

As always, the dispersion has to be estimated:

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

We proceed to fit a linear model and test for the treatment effect. Note that we can omit the `coef` argument to `glmLRT` because the treatment effect is the last coefficient in the model.

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

This test detects genes that are differentially expressed in response to the active treatment compared to the control, adjusting for baseline differences between the patients. This test can be viewed as a generalization of a paired *t*-test.

See the oral carcinomas case study of Section 4.4 for a fully worked analysis with paired samples.

3.4.2 Blocking

Paired samples are a simple example of what is called "blocking" in experimental design. The idea of blocking is to compare treatments using experimental subjects that are as similar as possible, so that the treatment different stands out as clearly as possible.

Suppose for example that we wish to compare three treatments A, B and C using experimental animals. Suppose that animals from the same litter are appreciably more similar than animals from different litters. This might lead to an experimental setup like:

FileName	Litter	Treatment
File1	1	A
File2	1	B
File3	1	C
File4	2	B
File5	2	A
File6	2	C
File7	3	C
File8	3	B
File9	3	A

Here it is the differences between the treatments that are of interest. The differences between the litters are not of primary interest, nor are we interested in a treatment effect that occurs for in only one litter, because that would not be reproducible.

We can compare the three treatments adjusting for any baseline differences between the litters by fitting an additive model:

```
> Litter <- factor(targets$Litter)
> Treatment <- factor(targets$Treatment)
> design <- model.matrix(~Litter+Treatment)
```

This creates a design matrix with five columns: three for the litters and two more for the differences between the treatments.

If `fit` is the fitted model with this design matrix, then we may proceed as follows. To detect genes that are differentially expressed between any of the three treatments, adjusting for litter differences:

```
> lrt <- glmLRT(fit, coef=4:5)
> topTags(lrt)
```

To detect genes that are differentially expressed in treatment B vs treatment A:

```
> lrt <- glmLRT(fit, coef=4)
> topTags(lrt)
```

To detect genes that are differentially expressed in treatment C vs treatment A:

```
> lrt <- glmLRT(fit, coef=5)
> topTags(lrt)
```

To detect genes that are differentially expressed in treatment C vs treatment B:

```
> lrt <- glmLRT(fit, contrast=c(0,0,0,-1,1))
> topTags(lrt)
```

The advantage of using litter as a blocking variable in the analysis is that this will make the comparison between the treatments more precise, if litter-mates are more alike that animals from different litters. On the other hand, if litter-mates are no more alike than animals from different litters, which might be so for genetically identical inbred laboratory animals, then the above analysis is somewhat inefficient because the litter effects are being estimated unnecessarily. In that case, it would be better to omit litter from the model formula.

3.4.3 Batch Effects

Another situation in which additive model formulas are used is when correcting for batch effects in an experiment. The situation here is analogous to blocking, the only difference being that the batch effects were probably unintended rather than a deliberate aspect of the experimental design. The analysis is the same as for blocking. The treatments can be adjusted for differences between the batches by using an additive model formula of the form:

```
> design <- model.matrix(~Batch+Treatment)
```

In this type of analysis, the treatments are compared only within each batch. The analysis is corrected for baseline differences between the batches.

The Arabidopsis case study in Section 4.5 gives a fully worked example with batch effects.

3.5 Comparisons Both Between and Within Subjects

Here is a more complex scenario, posed by a poster to the Bioconductor mailing list. The experiment has 18 RNA samples collected from 9 subjects. The samples correspond to cells from 3 healthy patients, either treated or not with a hormone; cells from 3 patients with disease 1, either treated or not with the hormone; and cells from 3 patients with disease 2, either treated or not with the hormone. The targets frame looks like this:

```
> targets
    Disease Patient Treatment
1   Healthy       1      None
2   Healthy       1   Hormone
3   Healthy       2      None
4   Healthy       2   Hormone
5   Healthy       3      None
6   Healthy       3   Hormone
7  Disease1       4      None
8  Disease1       4   Hormone
9  Disease1       5      None
10 Disease1       5   Hormone
11 Disease1       6      None
12 Disease1       6   Hormone
13 Disease2       7      None
14 Disease2       7   Hormone
15 Disease2       8      None
16 Disease2       8   Hormone
17 Disease2       9      None
18 Disease2       9   Hormone
```

If all the RNA samples were collected from independent subjects, then this would be nested factorial experiment, from which we would want to estimate the treatment effect for each disease group. As it is, however, we have a paired comparison experiment for each disease group. The feature that makes this experiment complex is that some comparisons (between the diseases) are made *between* patients while other comparisons (hormone treatment vs no treatment) are made *within* patients.

The design matrix will be easier to construct in R if we re-number the patients within each disease group:

```
> Patient <- gl(3,2,length=18)
```

We also define Disease and Treatment to be factors, with the control state as the first level in each case:

```
> Disease <- factor(targets$Disease, levels=c("Healthy","Disease1","Disease2"))
> Treatment <- factor(targets$Treatment, levels=c("None","Hormone"))
```

This gives us a revised targets frame:

```
> data.frame(Disease,Patient,Treatment)
    Disease Patient Treatment
1   Healthy       1      None
2   Healthy       1   Hormone
3   Healthy       2      None
4   Healthy       2   Hormone
5   Healthy       3      None
6   Healthy       3   Hormone
7  Disease1       1      None
8  Disease1       1   Hormone
9  Disease1       2      None
10 Disease1       2   Hormone
11 Disease1       3      None
12 Disease1       3   Hormone
13 Disease2       1      None
14 Disease2       1   Hormone
15 Disease2       2      None
16 Disease2       2   Hormone
17 Disease2       3      None
18 Disease2       3   Hormone
```

Now we can construct the design matrix. The critical feature to appreciate is that Patient and Treatment are of interest within each disease group, so we use the nested factorial formula discussed in a previous section. The patients are nested with the disease groups, because we have different patients in each group. The treatment is nested within disease groups, because we are interested in the disease-specific treatment effects. The model formula has the main effect for disease plus nested interactions with Patient and Treatment:

```
> design <- model.matrix(~Disease+Disease:Patient+Disease:Treatment)
> colnames(design)
 [1] "(Intercept)"                      "DiseaseDisease1"
 [3] "DiseaseDisease2"                  "DiseaseHealthy:Patient2"
 [5] "DiseaseDisease1:Patient2"         "DiseaseDisease2:Patient2"
 [7] "DiseaseHealthy:Patient3"          "DiseaseDisease1:Patient3"
 [9] "DiseaseDisease2:Patient3"         "DiseaseHealthy:TreatmentHormone"
[11] "DiseaseDisease1:TreatmentHormone" "DiseaseDisease2:TreatmentHormone"
```

After estimating the dispersions (code not shown), we can fit a linear model:

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

To find genes responding to the hormone in healthy patients:

```
> lrt <- glmLRT(fit, coef="DiseaseHealthy:TreatmentHormone")
> topTags(lrt)
```

To find genes responding to the hormone in disease1 patients:

```
> lrt <- glmLRT(fit, coef="DiseaseDisease1:TreatmentHormone")
> topTags(lrt)
```

To find genes responding to the hormone in disease2 patients:

```
> lrt <- glmLRT(fit, coef="DiseaseDisease2:TreatmentHormone")
> topTags(lrt)
```

To find genes that respond to the hormone in *any* disease group:

```
> lrt <- glmLRT(fit, coef=10:12)
> topTags(lrt)
```

To find genes that respond differently to the hormone in disease1 vs healthy patients:

```
> lrt <- glmLRT(fit, contrast=c(0,0,0,0,0,0,0,0,0,-1,1,0))
> topTags(lrt)
```

To find genes that respond differently to the hormone in disease2 vs healthy patients:

```
> lrt <- glmLRT(fit, contrast=c(0,0,0,0,0,0,0,0,0,-1,0,1))
> topTags(lrt)
```

To find genes that respond differently to the hormone in disease2 vs disease1 patients:

```
> lrt <- glmLRT(fit, contrast=c(0,0,0,0,0,0,0,0,0,0,-1,1))
> topTags(lrt)
```

Chapter 4

Case studies

4.1 SAGE profiles of normal and tumour tissue

4.1.1 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 epithelial cells. Two tumor and two normal RNA samples were available from different individuals.

4.1.2 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
```

4.1.3 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 samples. 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 expression 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
```

4.1.4 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. Although we do see some differences between the tumour samples, which are noticeably different from one another when compared against the normals, which are very similar to each other.

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

4.1.5 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 of 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
```

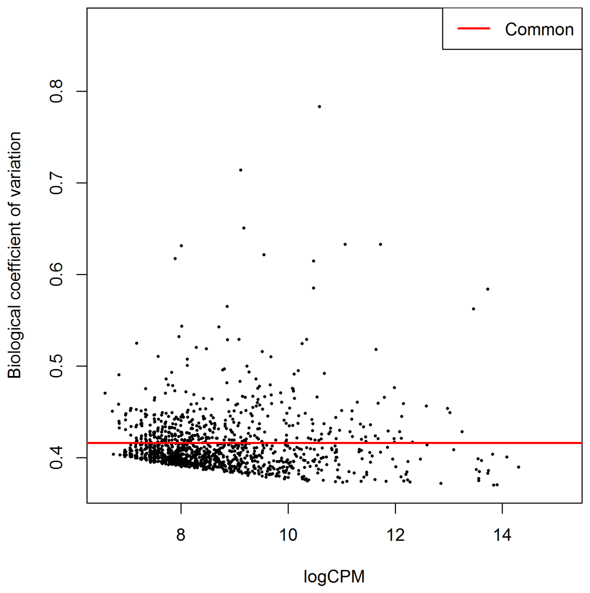
The square root of the common dispersion gives the coefficient of variation of biological variation (BCV). Here the BCV is 41%. This is a relatively large value, but not atypical for observational studies on human tumor tissue where the replicates are independent tumors or individuals.

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)
```



4.1.6 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
```

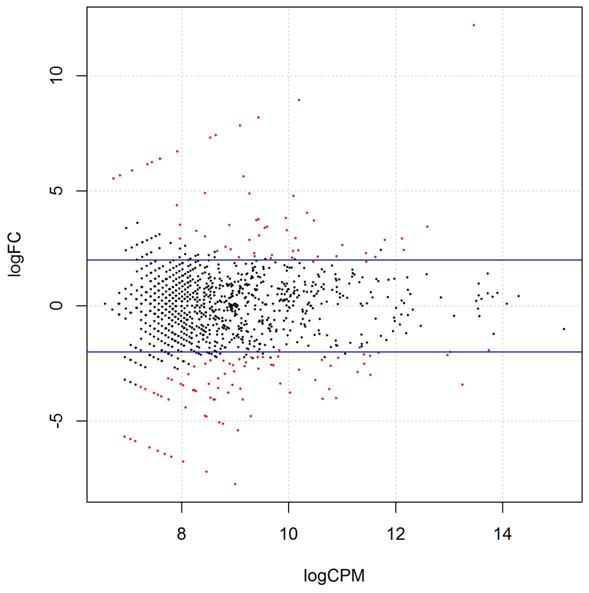
The total number of differentially expressed genes at FDR< 0.05 is:

```
> 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-regulated, non-differentially expressed and up-regulated 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.

4.1.7 Setup

This analysis was conducted on:

```
> sessionInfo()
R version 2.15.2 (2012-10-26)
Platform: i386-w64-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_3.0.6  limma_3.15.5

loaded via a namespace (and not attached):
[1] tools_2.15.2
```

4.2 deepSAGE of wild-type vs Dclk1 transgenic mice

4.2.1 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 over-expressing 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 the Solexa/Illumina 1G Genome Analyzer greatly to increase the number of tags that can be sequenced.

The RNA samples came from wild-type male C57/BL6j mice and transgenic mice over-expressing 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.

4.2.2 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"                "GSM272106.txt"
 [4] "GSM272318.txt"                "GSM272319.txt"                "GSM272320.txt"
 [7] "GSM272321.txt"                "GSM272322.txt"                "GSM272323.txt"
[10] "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 norm.factors
DCLK1 GSM272105.txt  DCLK Dclk1 transgenic mouse hippocampus  2685418            1
WT1   GSM272106.txt    WT        wild-type mouse hippocampus  3517977            1
DCLK2 GSM272318.txt  DCLK Dclk1 transgenic mouse hippocampus  3202246            1
WT2   GSM272319.txt    WT        wild-type mouse hippocampus  3558260            1
DCLK3 GSM272320.txt  DCLK Dclk1 transgenic mouse hippocampus  2460753            1
WT3   GSM272321.txt    WT        wild-type mouse hippocampus   294909            1
DCLK4 GSM272322.txt  DCLK Dclk1 transgenic mouse hippocampus   651172            1
WT4   GSM272323.txt    WT        wild-type mouse hippocampus  3142280            1
> dim(d)
[1] 844316      8
```

4.2.3 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
```

Having filtered, reset the library sizes:

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

4.2.4 Normalization

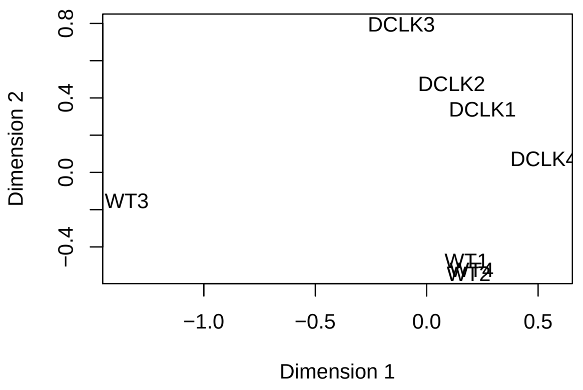
For this SAGE data, composition normalization is not so strongly required as for RNA-Seq data. Nevertheless, we align the upper-quartiles of the counts-per-million between the libraries:

```
> d <- calcNormFactors(d,method="upperquartile")
> d$samples
              files group                        description lib.size norm.factors
DCLK1 GSM272105.txt  DCLK Dclk1 transgenic mouse hippocampus  2441387        1.033
WT1   GSM272106.txt    WT        wild-type mouse hippocampus  3198460        0.979
DCLK2 GSM272318.txt  DCLK Dclk1 transgenic mouse hippocampus  2895690        1.051
WT2   GSM272319.txt    WT        wild-type mouse hippocampus  3210704        0.975
DCLK3 GSM272320.txt  DCLK Dclk1 transgenic mouse hippocampus  2225219        1.016
WT3   GSM272321.txt    WT        wild-type mouse hippocampus   271817        0.960
DCLK4 GSM272322.txt  DCLK Dclk1 transgenic mouse hippocampus   601062        1.013
WT4   GSM272323.txt    WT        wild-type mouse hippocampus  2855960        0.975
```

4.2.5 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:

```
> plotMDS(d)
```



The DCLK and WT samples separate quite nicely.

4.2.6 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 = 0.152 , BCV = 0.39
```

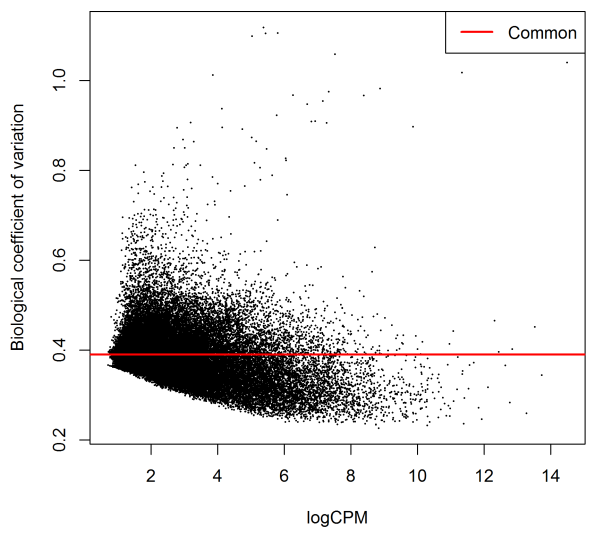
The biological coefficient of variation is the square root of the common dispersion.

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 it is not usually required for SAGE data:

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

The following plot displays the estimates:

```
> plotBCV(d)
```



4.2.7 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  9.33   5.46 5.40e-19 2.42e-14
CATAAGTCACAGAGTCG  9.78   3.58 4.28e-18 9.62e-14
CCAAGAATCTGGTCGTA  3.83   3.65 1.78e-14 2.66e-10
ATACTGACATTTCGTAT -4.40   4.49 1.24e-13 1.16e-09
GCTAATAAATGGCAGAT  3.11   5.92 1.29e-13 1.16e-09
CTGCTAAGCAGAAGCAA  3.34   3.91 1.59e-13 1.19e-09
AAAAGAAATCACAGTTG  9.45   3.17 2.10e-13 1.35e-09
TTCCTGAAAATGTGAAG  3.57   3.92 6.05e-13 3.39e-09
TATTTTGTTTTGTCGTA -4.04   4.08 2.91e-12 1.45e-08
CTACTGCAGCATTATCG  2.95   4.06 7.61e-12 3.42e-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)]
```

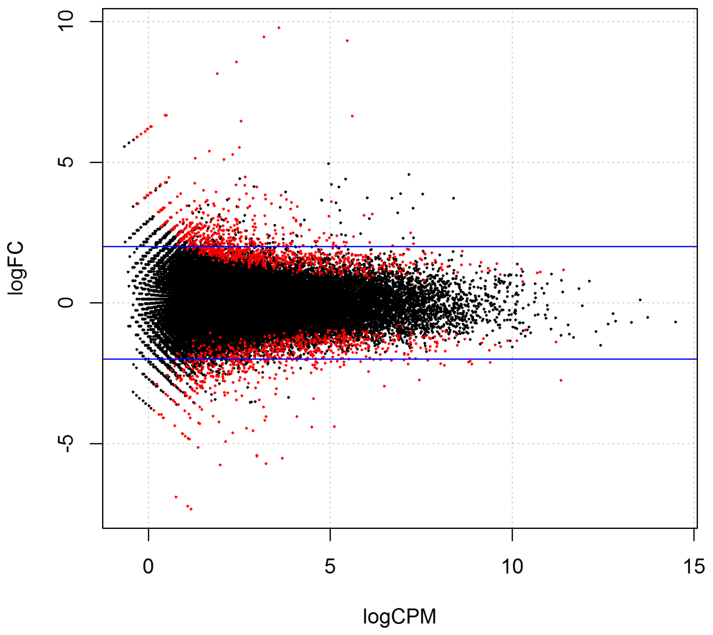
```
                  DCLK1  DCLK2  DCLK3  DCLK4    WT1    WT2   WT3   WT4
TCTGTACGCAGTCAGGC 65.54  34.88 197.73  54.90  0.000  0.311  0.00  0.00
CATAAGTCACAGAGTCG 27.44  26.59  26.06  11.65  0.000  0.000  0.00  0.00
CCAAGAATCTGGTCGTA 28.67  22.79  21.12  21.63  0.938  1.557  0.00  2.45
ATACTGACATTTCGTAT  2.05   1.73   3.60   1.66 35.330 71.012 14.72 36.42
GCTAATAAATGGCAGAT 158.52 110.85  59.32 118.12 14.069  9.967  3.68 13.31
CTGCTAAGCAGAAGCAA 31.13  30.39  23.37  24.96  2.189  2.180  0.00  3.85
AAAAGAAATCACAGTTG 12.70  31.08  18.87   4.99  0.000  0.000  0.00  0.00
TTCCTGAAAATGTGAAG 30.31  24.17  38.65  16.64  1.876  2.803  0.00  2.45
TATTTTGTTTTGTCGTA  4.10   1.73   1.35   0.00 27.513 53.259 14.72 23.46
CTACTGCAGCATTATCG 30.31  33.50  31.46  21.63  3.752  3.426  0.00  4.55
```

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

```
> summary(de <- decideTestsDGE(et, p=0.05))
   [,1]
-1   685
0  43313
1    884
```

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.

4.2.8 Setup

This analysis was conducted on:

```
> sessionInfo()
R version 2.15.2 (2012-10-26)
Platform: i386-w64-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_3.0.6  limma_3.15.5

loaded via a namespace (and not attached):
[1] tools_2.15.2
```

4.3 Androgen-treated prostate cancer cells (RNA-Seq, two groups)

4.3.1 Introduction

This case study considers RNA-Seq data from a treatment vs control experiment with relatively low biological variability.

4.3.2 RNA Samples

Genes stimulated by androgens (male hormones) are implicated in the survival of prostate cancer cells and are potential target of anti-cancer treatments. Three replicate RNA samples were collected from prostate cancer cells (LNCaP cell line) after treatment with an androgen hormone (100uM of DHT). Four replicate control samples were also collected from cells treated with an inactive compound [Li et al., 2008].

4.3.3 Sequencing

35bp reads were sequenced on an Illumina 1G Genome Analyzer using seven lanes of one flow-cell. FASTA format files are available from `http://yeolab.ucsd.edu/yeolab/Papers.html`.

4.3.4 Read mapping

Reads were mapped and summarized at the gene level as previously described by Young et al. [2010]. Reads were mapped to the NCBI36 build of the human genome using Bowtie, allowing up to two mismatches. Reads not mapping uniquely were discarded. The number of reads overlapping the genomic span of each Ensembl gene (version 53) was counted. Reads mapping to introns and non-coding regions were included. The tab-delimited file of read counts can be downloaded as `pnas_expression.txt` from `http://sites.google.com/site/davismcc/useful-documents`.

4.3.5 Reading the data

Read the targets file associating treatments with samples:

```
> targets <- readTargets()
> targets
     Lane Treatment Label
Con1    1   Control  Con1
Con2    2   Control  Con2
Con3    2   Control  Con3
Con4    4   Control  Con4
DHT1    5       DHT  DHT1
DHT2    6       DHT  DHT2
DHT3    8       DHT  DHT3
```

Read the file of counts:

```
> x <- read.delim("pnas_expression.txt", row.names=1, stringsAsFactors=FALSE)
> head(x)
                lane1 lane2 lane3 lane4 lane5 lane6 lane8  len
ENSG00000215696     0     0     0     0     0     0     0  330
ENSG00000215700     0     0     0     0     0     0     0 2370
ENSG00000215699     0     0     0     0     0     0     0 1842
ENSG00000215784     0     0     0     0     0     0     0 2393
ENSG00000212914     0     0     0     0     0     0     0  384
ENSG00000212042     0     0     0     0     0     0     0   92
```

Put the counts and other information into a DGEList object:

```
> y <- DGEList(counts=x[,1:7], group=targets$Treatment, genes=data.frame(Length=x[,8]))
> colnames(y) <- targets$Label
> dim(y)
[1] 37435     7
```

4.3.6 Filtering

We filter out very lowly expressed tags, keeping genes that are expressed at a reasonable level in at least one treatment condition. Since the smallest group size is three, we keep genes that achieve at least one count per million (cpm) in at least three samples:

```
> keep <- rowSums(cpm(y)>1) >= 3
> y <- y[keep,]
> dim(y)
[1] 16494     7
```

Re-compute the library sizes:

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

4.3.7 Normalizing

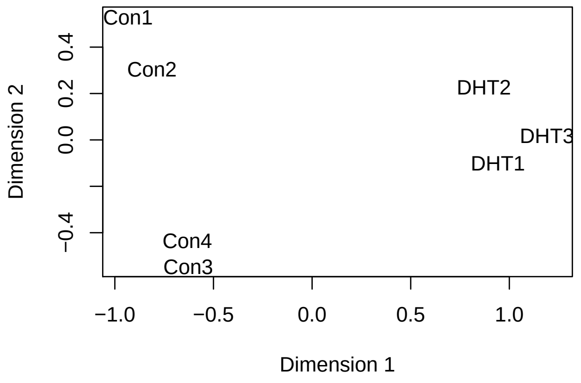
Compute effective library sizes using TMM normalization:

```
> y <- calcNormFactors(y)
> y$samples
       group lib.size norm.factors
Con1 Control   976847        1.030
Con2 Control  1154746        1.037
Con3 Control  1439393        1.036
Con4 Control  1482652        1.038
DHT1     DHT  1820628        0.954
DHT2     DHT  1831553        0.953
DHT3     DHT   680798        0.958
```

4.3.8 Data exploration

An MDS plots shows distances, in terms of biological coefficient of variation (BCV), between samples:

```
> plotMDS(y)
```



Dimension 1 clearly separates the control from the DHT-treated samples. This shows that the replicates are consistent, and we can expect to find lots of DE genes.

4.3.9 Estimating the dispersion

The common dispersion estimates the overall BCV of the dataset, averaged over all genes:

```
> y <- estimateCommonDisp(y, verbose=TRUE)
Disp = 0.02 , BCV = 0.141
```

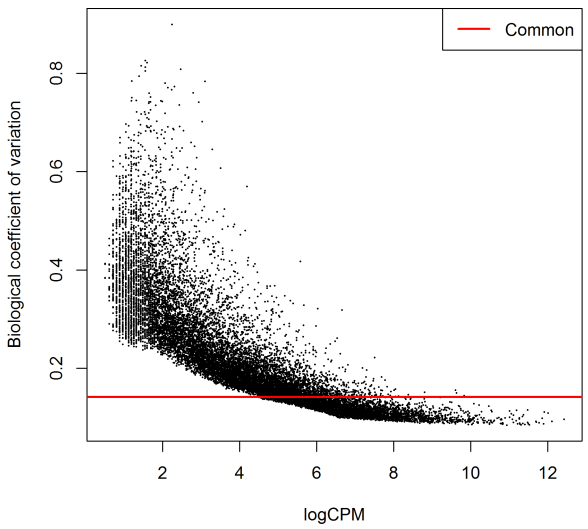
The BCV (square root of the common dispersion) here is 14%, a typical size for a laboratory experiment with a cell line or a model organism.

Now estimate gene-specific dispersions:

```
> y <- estimateTagwiseDisp(y)
```

Plot the estimated dispersions:

```
> plotBCV(y)
```



4.3.10 Differential expression

Compute exact genewise tests for differential expression between androgen and control treatments:

```
> et <- exactTest(y)
> top <- topTags(et)
> top
Comparison of groups:  DHT-Control
                Length logFC logCPM    PValue       FDR
ENSG00000151503   5605  5.82   9.71  0.00e+00  0.00e+00
ENSG00000096060   4093  5.00   9.94  0.00e+00  0.00e+00
ENSG00000166451   1556  4.66   8.83 1.15e-228 6.31e-225
ENSG00000127954   3919  8.17   7.20 1.00e-209 4.14e-206
ENSG00000162772   1377  3.32   9.74 2.09e-182 6.91e-179
ENSG00000113594  10078  4.08   8.03 5.07e-153 1.39e-149
ENSG00000116133   4286  3.26   8.78 6.33e-148 1.49e-144
ENSG00000115648   2920  2.63  11.47 2.82e-139 5.81e-136
ENSG00000123983   4305  3.59   8.58 8.38e-138 1.54e-134
ENSG00000116285   3076  4.22   7.35 1.05e-135 1.73e-132
```

Check the individual cpm values for the top genes:

```
> cpm(y)[rownames(top), ]
                 Con1  Con2   Con3   Con4 DHT1 DHT2 DHT3
ENSG00000151503  35.8  30.3  34.04  39.79 1816 1878 1798
ENSG00000096060  66.5  68.4  72.95  76.21 2183 2035 2131
ENSG00000166451  42.0  45.0  39.60  38.44  961  903 1069
ENSG00000127954   0.0   0.0   2.08   2.02  333  329  323
```

```
ENSG00000162772 176.1 176.7 173.68 205.04 1632 1785 1633
ENSG00000113594  37.9  31.2  39.60  29.00  514  524  614
ENSG00000116133  99.3  92.7 106.99  96.45  896  880  815
ENSG00000115648 962.3 938.7 914.97 907.16 5344 5428 4806
ENSG00000123983  63.5  65.8  65.31  72.84  744  687  922
ENSG00000116285  18.4  24.2  15.98  21.58  354  344  320
```

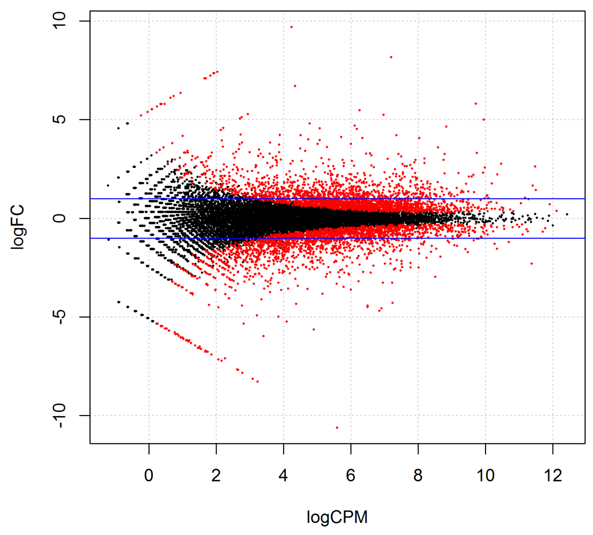
The total number of DE genes at 5% FDR is given by

```
> summary(de <- decideTestsDGE(et))
   [,1]
-1  2083
0  12127
1   2284
```

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

Plot the log-fold-changes, highlighting the DE genes:

```
> detags <- rownames(y)[as.logical(de)]
> plotSmear(et, de.tags=detags)
> abline(h=c(-1, 1), col="blue")
```



The blue lines indicate 2-fold changes.

4.3.11 Setup

The analysis of this section was conducted with:

```
> sessionInfo()
R version 2.15.2 (2012-10-26)
Platform: i386-w64-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_3.0.6  limma_3.15.5

loaded via a namespace (and not attached):
[1] tools_2.15.2
```

4.3.12 Acknowledgements

Thanks to Matthew Young for the file of read counts.

4.4 RNA-Seq of oral carcinomas vs matched normal tissue

4.4.1 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]. The aim of the analysis is to detect genes differentially expressed between tumor and normal tissue, adjusting for any differences between the patients. 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].

4.4.2 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])
```

4.4.3 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] 15578     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
```

4.4.4 Filtering

Different RefSeq transcripts for the same gene symbol count predominantly 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] 10526
```

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
```

4.4.5 Normalization

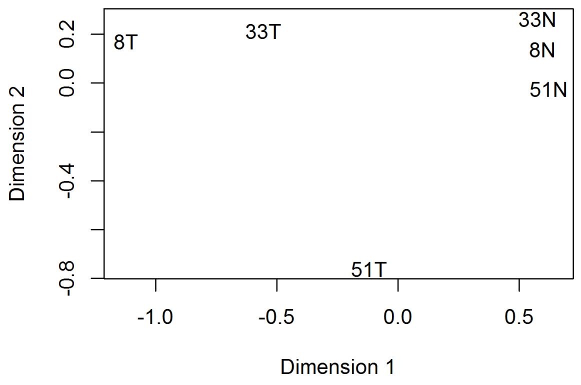
TMM normalization is applied to this dataset to account for compositional difference between the libraries.

```
> y <- calcNormFactors(y)
> y$samples
    group lib.size norm.factors
8N      1  7412036        1.154
8T      1  7137779        1.062
33N     1 15293080        0.656
33T     1 13691005        0.949
51N     1 19365708        1.089
51T     1 14422746        1.203
```

4.4.6 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 dimension 2 roughly corresponds to patient number. This confirms the paired nature of the samples. The tumor samples appear more heterogeneous than the normal samples.

4.4.7 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 expression 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.

4.4.8 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)
```

4.4.9 Differential expression

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

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

Conduct likelihood ratio tests for tumour vs normal tissue differences and show the top genes:

```
> lrt <- glmLRT(fit)
> topTags(lrt)
Coefficient:  TissueT
        RefSeqID    Symbol NbrOfExons logFC logCPM    LR   PValue      FDR
4118   NM_022440       MAL          2 -7.16   6.66 108.0 2.65e-25 2.79e-21
27179  NM_014440     IL36A          4 -6.14   5.48 103.5 2.64e-24 1.39e-20
5837   NM_005609      PYGM         20 -5.48   6.07  96.0 1.15e-22 4.03e-19
5737   NM_000959     PTGFR          3 -5.21   4.81  90.5 1.81e-21 4.75e-18
132724 NM_182502 TMPRSS11B         10 -7.41   7.72  87.7 7.62e-21 1.60e-17
4606   NM_004533    MYBPC2         28 -5.46   6.57  86.7 1.28e-20 2.24e-17
487    NM_173201    ATP2A1         22 -4.62   6.03  82.8 8.85e-20 1.33e-16
3850   NM_057088      KRT3          9 -5.83   6.57  81.5 1.80e-19 2.36e-16
2027   NM_053013      ENO3         12 -5.17   6.39  79.0 6.17e-19 7.22e-16
11240  NM_007365     PADI2         16 -4.56   6.41  71.8 2.35e-17 2.47e-14
```

Note that `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 are for tumor vs normal differential expression, adjusting for baseline differences between the three patients. (The tests can be viewed as analogous to paired t-tests.) The top DE tags have tiny p-values and FDR values, as well as large fold changes.

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
4118   279.7 0.140 192.0  7.012  69.8  0.4853
27179   49.5 1.401 119.3  3.287  41.4  0.0693
5837   188.7 3.082  82.8  1.169 112.1  7.1415
5737    61.4 0.981  18.5  0.876  89.6  3.1201
132724 349.7 0.420 510.4 23.446 174.1  0.6240
4606   130.3 1.541  31.8  0.438 415.4 31.6861
487    131.9 3.502 101.5  3.798 117.0 11.1629
3850   144.2 0.981 246.8 26.149  45.7  0.3467
2027   146.4 0.560  84.2  5.405 249.5 15.4617
11240  131.7 3.222  67.9  2.264 280.3 36.1928
```

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

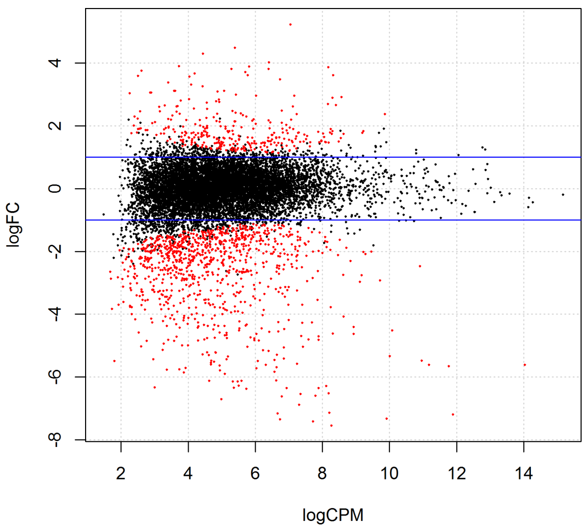
The total number of differentially expressed genes at 5% FDR is given by:

```
> summary(de <- decideTestsDGE(lrt))
```

```
   [,1]
-1  970
0  9251
1   305
```

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(-1, 1), col="blue")
```



The blue lines indicate 2-fold changes.

4.4.10 Setup

This analysis was conducted on:

```
> sessionInfo()
R version 2.15.2 (2012-10-26)
Platform: i386-w64-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_2.8.0   RSQLite_0.11.2       DBI_0.2-5
[4] AnnotationDbi_1.20.3 Biobase_2.18.0       BiocGenerics_0.4.0
[7] edgeR_3.0.6          limma_3.15.5

loaded via a namespace (and not attached):
[1] IRanges_1.16.4  parallel_2.15.2 stats4_2.15.2   tools_2.15.2
```

4.5 RNA-Seq of pathogen inoculated Arabidopsis with batch effects

4.5.1 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. The aim of the analysis therefore is to detect genes differentially expressed in response to ΔhrcC challenge, while correcting for any differences between the batches.

4.5.2 RNA samples

Pseudomonas syringae is a bacterium often used to study plant reactions to pathogens. In this experiment, six-week old Arabidopsis plants were 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.

4.5.3 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`).

4.5.4 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 are 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
```

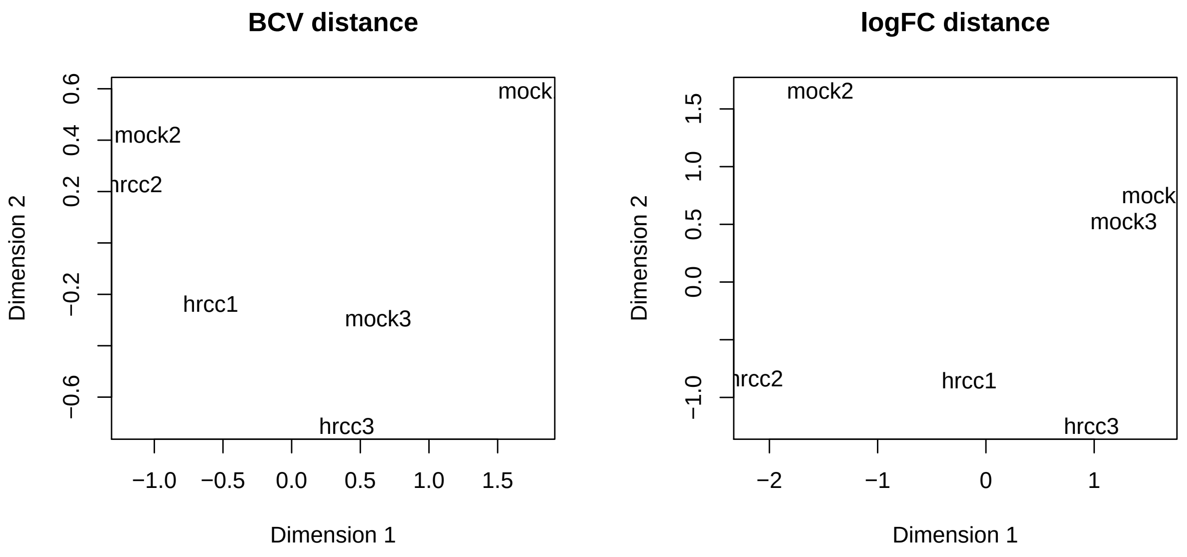
4.5.5 Data exploration

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

```
> logCPM <- predFC(y, prior.count=2*ncol(y))
> 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)
```

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.

4.5.6 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"
```

4.5.7 Estimating the dispersion

Estimate the average dispersion over all genes:

```
> y <- estimateGLMCommonDisp(y, design, verbose=TRUE)
Disp = 0.0705 , BCV = 0.266
```

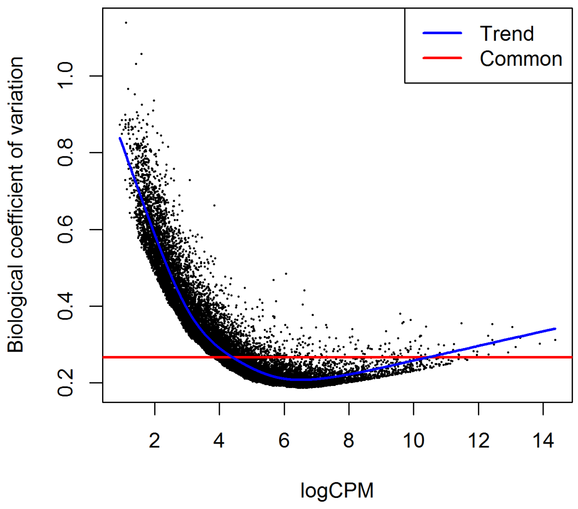
The square root of dispersion is the coefficient of biological variation (BCV). Here the common dispersion is 0.0705, so the BCV is 0.266. The common BCV is on the high side, considering that this is a designed experiment using genetically identical plants.

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

```
> y <- estimateGLMTrendedDisp(y, design)
> y <- estimateGLMTagwiseDisp(y, design, prior.df=10)
```

Here we have chosen `prior.df` slightly smaller than the default, which is 20, 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)
```



4.5.8 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(fit, coef=2:3)
> topTags(lrt)
Coefficient:  Time2 Time3
          logFC.Time2 logFC.Time3 logCPM  LR   PValue      FDR
AT5G66800        5.59      -1.075   5.43 274 2.53e-60 4.18e-56
AT5G31702        5.84      -2.612   5.90 237 4.33e-52 3.58e-48
AT5G23000        5.60      -0.292   5.68 235 1.18e-51 6.50e-48
AT3G33004        4.82      -1.770   5.59 227 6.44e-50 2.66e-46
```

```
AT2G45830        5.43      -0.597   4.64 190 6.11e-42 2.02e-38
AT2G11230        3.50      -1.534   5.56 174 2.09e-38 5.75e-35
AT5G35736        5.41      -1.013   4.55 162 5.69e-36 1.34e-32
AT2G08986        7.20      -0.563   6.81 161 1.11e-35 2.30e-32
AT2G07782        3.49      -1.620   5.23 161 1.30e-35 2.38e-32
AT2G18193        3.06      -2.407   5.02 147 1.30e-32 2.14e-29
> FDR <- p.adjust(lrt$table$PValue, method="BH")
> sum(FDR < 0.05)
[1] 3179
```

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(fit)
> topTags(lrt)
Coefficient:  Treathrcc
          logFC logCPM  LR   PValue      FDR
AT5G48430  6.34   6.71 257 7.07e-58 1.17e-53
AT2G19190  4.50   7.37 230 6.19e-52 5.11e-48
AT2G39530  4.34   6.70 212 5.63e-48 3.10e-44
AT3G46280  4.78   8.09 198 5.16e-45 2.13e-41
AT2G39380  4.95   5.75 189 4.16e-43 1.37e-39
AT1G51800  3.97   7.70 181 2.62e-41 7.21e-38
AT1G51820  4.34   6.36 172 3.44e-39 7.76e-36
AT1G51850  5.33   5.39 171 3.75e-39 7.76e-36
AT2G44370  5.43   5.17 164 1.70e-37 3.13e-34
AT3G55150  5.80   4.86 155 1.57e-35 2.60e-32
```

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)]
           mock1 mock2 mock3 hrcc1 hrcc2 hrcc3
AT5G48430  4.218  4.74  0.00 198.6 344.7 116.6
AT2G19190 16.343 12.64 12.00 358.6 279.1 327.3
AT2G39530  6.854  9.48 12.00 166.1 210.3 226.7
AT3G46280 18.452 17.91 16.62 404.4 410.3 765.3
AT2G39380  2.109  3.16  4.31  96.3  92.5 126.0
AT1G51800 28.469 17.38 27.70 380.8 381.0 432.6
AT1G51820  9.490  7.90  5.54 127.4 171.6 178.3
AT1G51850  1.054  1.05  3.39  82.1  61.7 101.5
AT2G44370  2.109  1.05  1.54  59.9  73.5  80.2
AT3G55150  0.527  1.05  1.23  45.3  71.2  60.0
```

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

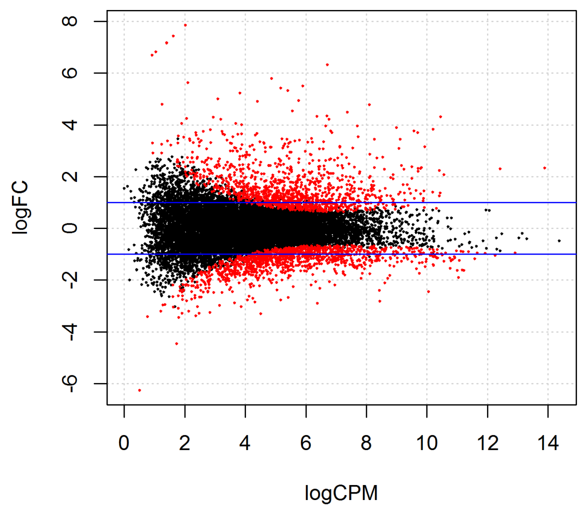
```
> summary(dt <- decideTestsDGE(lrt))
   [,1]
-1  1238
0  14038
1   1250
```

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.

4.5.9 Setup

This analysis was conducted on:

```
> sessionInfo()
R version 2.15.2 (2012-10-26)
Platform: i386-w64-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_3.0.6   limma_3.15.5  NBPSeq_0.1.8  qvalue_1.32.0

loaded via a namespace (and not attached):
[1] tcltk_2.15.2 tools_2.15.2
```

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`.

Kasper D. Hansen, Rafael A. Irizarry, and Zhijin WU. Removing technical variability in rna-seq data using conditional quantile normalization. *Biostatistics*, 13(2):204–216, 2012. doi: 10.1093/biostatistics/kxr054. URL `http://biostatistics.oxfordjournals.org/content/13/2/204.abstract`.

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*, 40(10):4288–4297, 2012. URL `http://nar.oxfordjournals.org/content/40/10/4288`.

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`.

Davide Risso, Katja Schwartz, Gavin Sherlock, and Sandrine Dudoit. GC-content normalization for RNA-Seq data. *BMC Bioinformatics*, 12:480, 2011.

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.

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`.

Matthew D. Young, Matthew J. Wakefield, Gordon K. Smyth, and Alicia Oshlack. Gene ontology analysis for RNA-seq: accounting for selection bias. *Genome Biology*, 11:R14, 2010.

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.