edgeR: differential expression analysis of digital gene expression data

User's Guide

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

8 September 2011

Contents

1 Introduction

This guide provides an overview of the capabilities of the Bioconductor package edgeR [Robinson et al., 2010], including a number of fully worked case studies. edgeR provides statistical routines for assessing differential expression or differential marking from sequencing data. The package can be applied to any technology that produces read counts for genomic features, but it is particularly intended for RNA-Seq, SAGE-Seq or ChIP-Seq data arising from sequencing technologies that produce short reads, including as Illumina^{TM}, 454 or ABI SOLiD technologies.

The package implements exact statistical methods for multigroup experiments developed by Robinson and Smyth [2007, 2008]. It also implements statistical methods based on generalized linear models, suitable for multifactor experiments of any complexity, developed by McCarthy et al. [2011]. Sometimes we refer to the former exact methods as *classic* edgeR, and the latter as *glm* edgeR. However the two sets of methods are complementary and can often be combined in the course of a data analysis. Most of the glm functions can be identified by the letters "`glm`" as part of the function name.

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

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

2 How to get help

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

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

3 Quick start

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

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

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

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

Many variants are available on this analysis.

4 Negative binomial models

The starting point for an RNA-Seq experiment is a set of n RNA samples, typically associated with a variety of treatment conditions. Each sample is sequenced, short reads are mapped to the appropriate genome, and the number of reads mapped to each genomic feature of interest is recorded. The number of reads from sample i mapped to gene g will be denoted y_{gi}. The set of genewise counts for sample i makes up the expression profile or *library* for that sample. The expected size of each count is the product of the library size and the relative abundance of that gene in that sample.

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

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

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

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

5 Reading data

edgeR requires three pieces of information:

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

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

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

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

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

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

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

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

6 Normalization issues for counts data

6.1 General comments

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

It bears emphasizing that RPKM values should *not* be used for assessing differential expression of genes between samples in edgeR. We use the raw counts, because the methods implemented in edgeR are based on the negative binomial distribution, a discrete distribution. To be able to perform good inference on differential expression it is very important to model the mean-variance relationship in the data appropriately. There are good reasons why the NB model is appropriate for the raw count data, but transforming the data using

RPKM (or FPKM or similar) renders our distributional assumptions invalid and we cannot guarantee that our methods will be reliable for such transformed data.

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

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

6.2 Calculating normalization factors

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

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

```
> head(D)
```

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

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

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

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

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

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

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

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

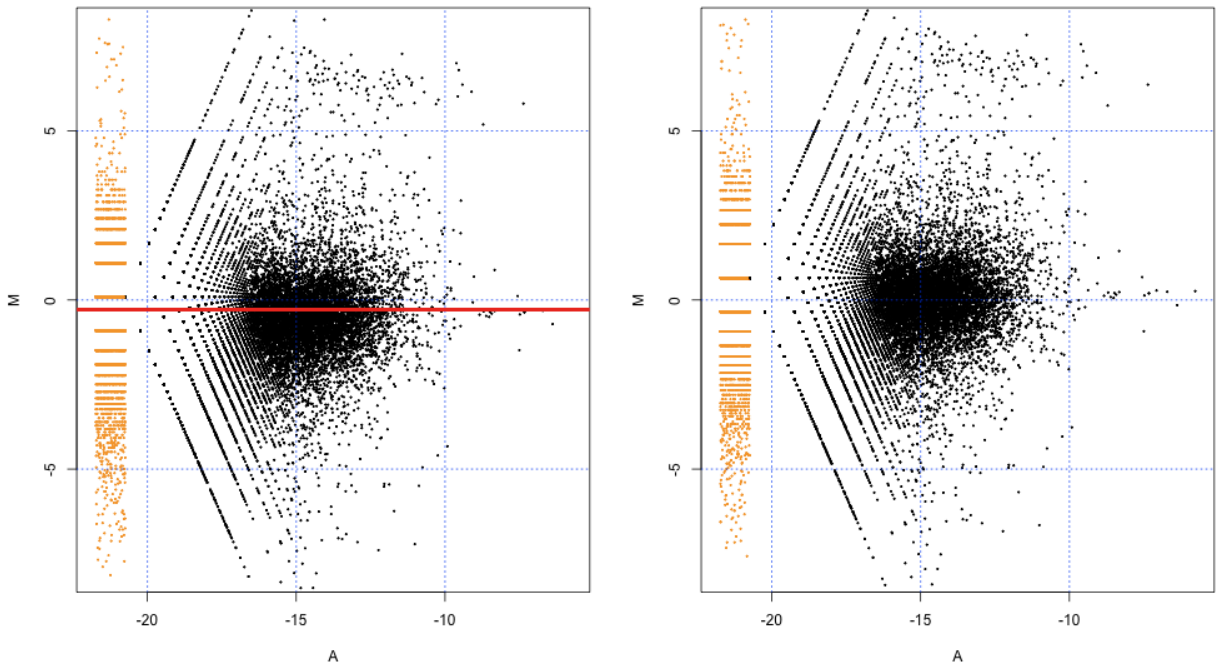

Figure 1: Smear plots before (left) and after (right) composition normalization.

7 Pairwise comparisons between group (classic)

7.1 Estimating dispersions

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

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

The qCML method can estimate a common dispersion for all the tags or separate dispersions for individual tags. As individual tags typically don't provide enough data to estimate the dispersion reliably, we implement an empirical Bayes strategy for squeezing the tagwise dispersions towards the common dispersion. The amount of shrinkage is determined by the

prior weight given to the common dispersion and the precision of the tagwise estimates. The prior can be thought of arising from a number of prior observations, equivalent to `prior.n` tags with common dispersion and the same number of libraries per tag as in the current experiment. The prior number of tags `prior.n` can be set by the user. The precision of the tagwise estimators is roughly proportion to the per-tag degrees of freedom, equal to the number of libraries minus the number of groups or the number of GLM coefficients. We generally recommend choosing `prior.n` so that the total degrees of freedom (`prior.n*df`) associated with the prior is about 20–30, subject to `prior.n` not going below 1. For example, if there are four libraries and two groups, the tagwise degrees of freedom are 2, so we would recommend `prior.n=10`. This is an empirical rule of thumb borne out of experience with a number of datasets.

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

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

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

To estimate common dispersion:

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

To estimate tagwise dispersions:

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

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

For more detailed examples, see the case studies in section 10 (Zhang data), section 11 ('t Hoen data), section 12 (Li data) and section 13 (Tuch data).

7.2 Testing for DE genes

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

The exact test is based on the qCML methods. Knowing the conditional distribution for the sum of counts in a group, we can compute exact p-values by summing over all sums

of counts that have a probability less than the probability under the null hypothesis of the observed sum of counts. The exact test for the negative binomial distribution has strong parallels with Fisher's exact test.

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

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

For more detailed examples, see the case studies in section 10 (Zhang's data), section 11 ('t Hoen's data) and section 12 (Li's data).

8 General experiments (glm functionality)

8.1 Estimating dispersions

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

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

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

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

To estimate common dispersion:

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

To estimate trended dispersions:

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

To estimate tagwise dispersions:

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

Note that we need to estimate either common dispersion or trended dispersions prior to the estimation of tagwise dispersions. When estimating tagwise dispersions, the empirical Bayes method is applied to shrink 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 13 (Tuch's data).

8.2 Testing for DE genes

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

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

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

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

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

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

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

To compare 2 vs 1:

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

To compare 3 vs 1:

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

To compare 3 vs 2:

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

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

For more detailed examples, see the case study in section 13 (Tuch's data).

9 What to do if you have no replicates

edgeR is primarily intended for use with data including biological replication. Nevertheless, RNA-Seq and ChIP-Seq are still expensive technologies, so it sometimes happens that only one library can be created for each treatment condition. In these cases there are no replicate libraries from which to estimate biological variability. In this situation, the data analyst is faced with the following choices:

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.

2. Remove one or more explanatory factors from the linear model in order to create some residual degrees of freedom. Estimate the dispersion from this reduced model, then test hypotheses about the full model using these dispersions. 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. This approach will only be successful if the number of DE genes is relatively small.

3. You could try

   ```
   estimateGLMCommonDisp(method="deviance",robust=TRUE,subset=NULL)
   ```

   This is our current best attempt at an automatic method to estimate dispersion without replicates. It will only give good results when the counts are not too small and the number of DE genes is not too large.

4. You could identify a number of transcripts that should not be DE, and estimate the dispersion from them:

   ```
   d0 <- estimateCommonDisp(d[housekeeping,])
   ```

5. Simply pick a reasonable dispersion value (`dispersion=0.3` say), based on your experience with similar data, and use that. Although subjective, this is much still better and more defensible than assuming Poission variation.

10 Case study: SAGE data

10.1 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].

10.2 Source of the data

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

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

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

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

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

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

In this case, the tag counts for the four individual libraries are stored in four separate plain text files, `GSM728.txt, GSM729.txt, GSM755.txt` and `GSM756.txt`. In each file, the tag IDs and counts for each tag are provided in a table. It is best to create a tab-delimited,

plain-text 'Targets' file, which, under the headings 'files', 'group' and 'description', gives the filename, the group and a brief description for each sample.

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

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

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

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

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

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

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

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

[1] 5012    4

> d$samples$lib.size
```

```
[1] 50179 49593 57686 49064

> colSums(d$counts)

    1     2     3     4
34970 35764 36940 30325

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

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

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

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

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

10.4 Analysis using common dispersion

10.4.1 Estimating the common dispersion

The first major step in the analysis of DGE data using the NB model is to estimate the dispersion parameter for each tag. The most straight-forward analysis of DGE data uses the common dispersion estimate as the dispersion for all tags. For many applications this will be adequate and it may not be necessary to estimate tagwise dispersions, i.e. estimate the dispersion parameter separately for each tag. Using the common dispersion allows the user to obtain DE results very quickly and in few steps, and so makes a good place to start with any analysis of DGE data.

Estimating the common dispersion is done using the function `estimateCommonDisp`. In order to do this, the function first needs to generate the 'pseudocounts' under the alternative hypothesis (that there really is a difference in expression level between the groups). The

conditional maximum likelihood method assumes that the library sizes are equal, which is certainly not true in general for DGE data.

The pseudocounts are calculated using a quantile-to-quantile method for the negative binomial distribution so that the library sizes for the pseudocounts are equal to the geometric mean of the original library sizes. These pseudocounts are then used as the count data for the common conditional negative binomial likelihood function, which is maximised over the dispersion parameter to obtain our estimate of the common dispersion.

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

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

The output of `estimateCommonDisp` is a DGEList object with several new elements. The element `common.dispersion`, as the name suggests, provides the estimate of the common dispersion, and `pseudo.alt` gives the pseudocounts calculated under the alternative hypothesis. The element `genes` contains the information about gene/tag identifiers. The element `conc` gives the estimates of the overall concentration of each tag across all of the original samples (`conc$conc.common`) and the estimate of the concentration of each tag within each group (`conc$conc.group`). The element `common.lib.size` gives the library size to which the original libraries have been adjusted in the pseudocounts.

```
> d$samples$lib.size

[1] 34970 35764 36940 30325

> d$common.lib.size

[1] 34404
```

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

```
> d$common.dispersion

[1] 0.197

> sqrt(d$common.dispersion)

[1] 0.444
```

10.4.2 Testing

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

```
> de.com <- exactTest(d)

Comparison of groups:  Tu - NC

> names(de.com)

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

> names(de.com$table)

[1] "logConc" "logFC"   "p.value"
```

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

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

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

```
> topTags(de.com)
```

```
Comparison of groups:  Tu-NC
           logConc  logFC  P.Value      FDR
AGCTGTTCCC   -28.0  44.13 7.65e-20 3.84e-16
CTTGGGTTTT   -29.6  40.86 2.61e-10 6.54e-07
TACAAAATCG   -30.0  40.10 2.79e-08 4.53e-05
CCCAACGCGC   -12.2  -5.77 3.62e-08 4.53e-05
GCCACCCCCT   -30.0  39.96 5.83e-08 5.85e-05
CCAGTCCGCC   -30.1  39.76 1.97e-07 1.63e-04
GTCATCACCA   -30.1 -39.73 2.28e-07 1.63e-04
CGCGTCACTA   -11.7   4.73 4.84e-07 2.75e-04
TCACCGGTCA   -10.5  -4.14 4.94e-07 2.75e-04
TAAATTGCAA   -10.8  -4.17 6.75e-07 3.38e-04
```

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

```
> detags.com <- rownames(topTags(de.com)$table)
> d$counts[detags.com, ]

             1  2   3    4
AGCTGTTCCC   0  0 119 1011
CTTGGGTTTT   0  0  21   97
TACAAAATCG   0  0  14   56
CCCAACGCGC 106  1   2    0
GCCACCCCCT   0  0   5   58
CCAGTCCGCC   0  0   6   49
GTCATCACCA  35 20   0    0
CGCGTCACTA   1  3  88   21
TCACCGGTCA 118 75   6    5
TAAATTGCAA 103 59   3    6
```

If we order the genes by fold change instead of p-value, as in the table below, we see that the genes with the largest fold changes have very small concentrations. This ranking is dominated by genes that have zero total counts in one group and is less informative than ranking by p-value.

```
> topTags(de.com, sort.by = "logFC")

Comparison of groups:  Tu-NC
           logConc logFC  P.Value      FDR
AGCTGTTCCC   -28.0  44.1 7.65e-20 3.84e-16
CTTGGGTTTT   -29.6  40.9 2.61e-10 6.54e-07
TACAAAATCG   -30.0  40.1 2.79e-08 4.53e-05
GCCACCCCCT   -30.0  40.0 5.83e-08 5.85e-05
CCAGTCCGCC   -30.1  39.8 1.97e-07 1.63e-04
GTCATCACCA   -30.1 -39.7 2.28e-07 1.63e-04
```

```
GTGCGCTGAG   -30.4  39.3 2.23e-06 7.46e-04
GTGTGTTTGT   -30.4  39.2 3.92e-06 1.03e-03
CTTGACATAC   -30.4 -39.2 3.92e-06 1.03e-03
GGGGGGGGGG   -30.4  39.1 4.77e-06 1.14e-03
```

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

We see in the output below that 243 genes are significantly differentially expressed according to edgeR when using the common dispersion estimate. Of those genes, 101 are up-regulated in the cancer cells compared with the normal cells and 142 are down-regulated in the cancer cells compared with normal cells. These proportions of up- and down-regulated tags are very similar to those found by Zhang et al. [1997].

```
> sum(de.com$table$p.value < 0.01)

[1] 243

> top243 <- topTags(de.com, n = 243)
> sum(top243$table$logFC > 0)

[1] 101

> sum(top243$table$logFC < 0)

[1] 142
```

Furthermore, we see below that 99 tags (2% of the total number of genes after filtering) have a p-value of less than 0.05 after adjusting for multiple testing using the Benjamini and Hochberg [1995] method for controlling the FDR, which is strong evidence for differential expression.

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

   [,1]
-1   57
0  4913
1    42
```

10.4.3 Visualising DGE results

The function `plotSmear` can be used to generate a plot of the log-fold change against the log-concentration for each tag (analogous to an MA-plot in the microarray context). We can easily identify the top DE tags and highlight them on the plot. The code for producing the default fold-change plot is shown below, and the result of this code is shown in Figure 2.

```
> detags243 <- rownames(top243$table)
> png(file="edgeR_case_study_Zhang_smearplot.png", height=600, width=600)
> plotSmear(d, de.tags = detags243, main = "FC plot using common dispersion")
> abline(h = c(-2, 2), col = "dodgerblue")
> dev.off()

null device
          1
```

Figure 2 shows the default fold change-plot for these data—the 'smear plot'. Plotting DGE data poses some challenges, as when the total counts in one group are zero, the log-fold change is technically infinite, and the log-concentration is negative infinity. With the algorithm used by topTags, we see very high log-fold changes and very small values for log-concentration for such tags, but plotting these values directly causes problems with the scale of the graph. To get around this problem, edgeR produces a 'smear' of points at the left-most edge of the plot for tags which have zero counts in one of the groups. Although this is still slightly artificial, it has the advantage that the expression level of all tags can be visualised and interpreted simultaneously.

The 'lines' of points we see at smaller log-concentration values arise from the discrete nature of the count data. When the sum all of the counts in one of the groups is one, we see the lines of points furthest away from the main body of points, and other lines of points correspond to when the total sum of counts in one of the groups is 2, 3, 4 and so on.

In Figure 2, the 264 tags identified as differentially expressed (i.e. those identified as significant (p-value less than 0.01) by edgeR using the common dispersion) are outlined in red.

10.5 Analysing the data using moderated tagwise dispersions

10.5.1 Moderating the tagwise dispersion

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

To run the moderated analysis, we need to determine how much moderation is necessary. As discussed above, we currently prefer to choose a sensible value for the smoothing parameter *a priori*, although we do have an algorithm developed by Robinson and Smyth [2007] for estimating the smoothing parameter as an approximate eBayes rule.

In our experience analysing RNA-Seq data we have found that a good rule of thumb for choosing a value for `prior.n` is to choose a certain number of prior degrees of freedom (a value between 20 and 30 works well) and then divide this number by the degrees of freedom (number of samples minus the number of variables being fit in the model; in the case of a

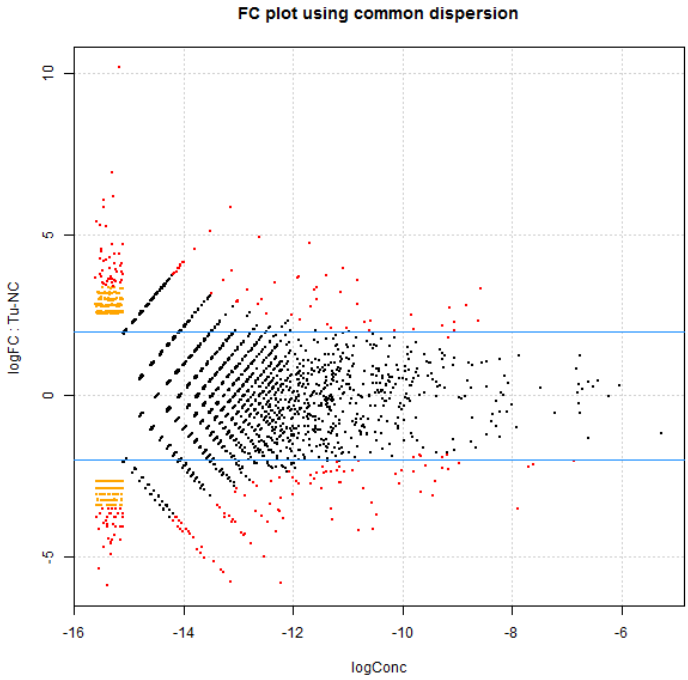

Figure 2: Plot of the log-fold change against the log-concentration for each tag. The 243 most differentially expressed tags as identified by edgeR using the common dispersion are outlined in red.

simple multiple-group comparison this is simply the number of samples minus the number of groups). The function `getPriorN` automatically calculates the appropriate value for `prior.n` for a given experimental design.

In this experiment we only have four libraries, a small sample size, so we should not be too confident about the accuracy of the tagwise dispersions. Here, we have just two samples in each group, and thus two degrees of freedom for estimating the dispersion parameter. Thus, setting the `prior.n` to be 10 (20 divided by two) should be appropriate. This means that the common likelihood receives the weight of ten individual tags. Therefore, there will be a reasonable degree of 'squeezing' towards the common dispersion estimate, but still enough scope to allow flexibility when estimating the individual dispersion for each gene. By default, `estimateTagwiseDisp` uses the `prior.n` value from `getPriorN`, which here recommends the value of 10 for `prior.n`..

The function `estimateTagwiseDisp` produces a DGEList object that contains all of the elements present in the object produced by `estimateCommonDisp`, as well as the value for `prior.n` used (`d$prior.n`) and the tagwise dispersion estimates (`d$tagwise.dispersion`), as we see below. Here we set `grid.length=500` for greater precision in the tagwise dispersion

estimates. We use the `trend` argument below to indicate that we wish to allow a dependence of the dispersion on the overall expression level.

```
> d <- estimateTagwiseDisp(d, trend=TRUE, prop.used=0.5, grid.length=500)

Using grid search to estimate tagwise dispersion.

> names(d)

 [1] "samples"            "common.dispersion"  "prior.n"
 [4] "tagwise.dispersion" "counts"             "pseudo.alt"
 [7] "genes"              "all.zeros"          "conc"
[10] "common.lib.size"

> head(d$tagwise.dispersion)

[1] 0.764 0.167 0.156 0.203 0.170 0.178
```

It is interesting to consider the distribution of the tagwise dispersion estimates. As we can see from the output below, the tagwise dispersion estimates range from a minimum of 0.08 to a maximum of 0.76. The range of dispersions is therefore large, but the tags in the middle two quartiles of the tagwise dispersion estimates have dispersion estimates close to the common dispersion estimate.

```
> summary(d$tagwise.dispersion)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.079   0.159   0.181   0.200   0.221   0.764

> d$common.dispersion

[1] 0.197

> png(file="edgeR_case_study_Zhang_tgwdisp_vs_abundance.png",
+     height=600, width=600)
> plot(log2(1e06*d$conc$conc.common), d$tagwise.dispersion,
+      xlab="Counts per million (log2 scale)", ylab="Tagwise dispersion")
> abline(h=d$common.dispersion, col="firebrick", lwd=3)
> dev.off()

null device
          1
```

Figure 3 shows a plot of the tagwise dispersion estimates against counts per million (i.e. tag abundance). We see that there are more tags at the lower-abundance end of the spectrum and that these tags tend to be more variable in terms of tagwise dispersion estimates (many tags have dispersion estimates much higher or lower than the common dispersion). For higher abundance levels, tagwise estimates tend to be quite close to the common estimate, apart from those handful of tags which are markedly more variable than the others.

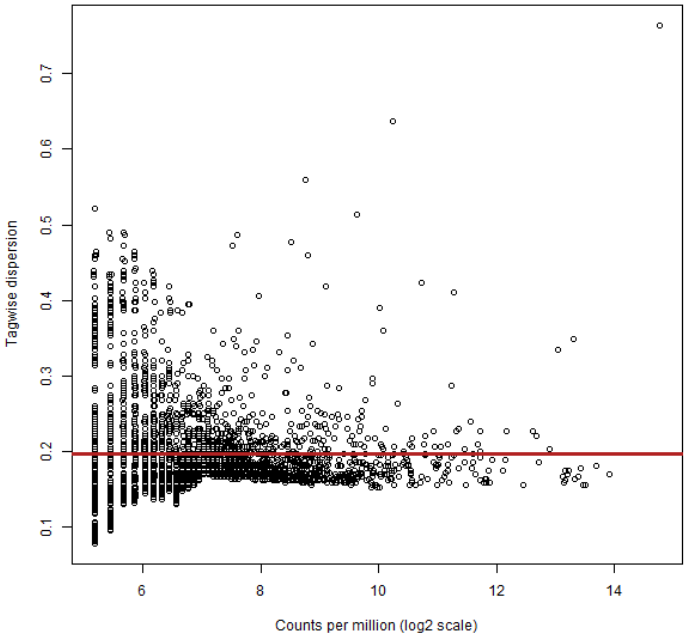

Figure 3: Plot of the tagwise dispersion estimates against the counts per million (on the log2 scale) for each tag. The red line shows the common dispersion estimate.

10.5.2 Testing

The testing procedures when using tagwise dispersion estimates are carried out exactly as for the common dispersion, as described above, but we add the argument `common.disp=FALSE` to the call to `exactTest`. Here we carry out the testing using the tagwise dispersion estimates calculated using a `prior.n` value of ten.

```
> de.tgw <- exactTest(d, common.disp = FALSE)

Comparison of groups:  Tu - NC
```

The output below shows that when using tagwise dispersions, the edgeR package still identifies a lot of differential expression between the normal colon cell group and the primary CR cancer cell group. This arises because the moderated tagwise dispersions can be much smaller or larger than the common dispersion, and tags with smaller dispersions will have smaller p-values than the same tags with p-values computed using a common dispersion (vice versa for tags with larger dispersions). As with the analysis using the common dispersion, all of the top tags have a large fold change, indicating that these changes in expression are likely to be biologically meaningful. We note that the ranking is different, however, and not

all of the top ten tags according to using the common dispersion are found to be among the top ten tags using tagwise dispersions.

```
> topTags(de.tgw)

Comparison of groups:  Tu-NC
           logConc  logFC  P.Value      FDR
AGCTGTTCCC   -28.0  44.13 2.91e-13 1.46e-09
CTTGGGTTTT   -29.6  40.86 1.22e-08 3.06e-05
TCACCGGTCA   -10.5  -4.14 1.05e-07 1.75e-04
GTCATCACCA   -30.1 -39.73 1.48e-07 1.86e-04
TACAAAATCG   -30.0  40.10 2.56e-07 2.57e-04
TAAATTGCAA   -10.8  -4.17 3.14e-07 2.62e-04
TAATTTTTGC   -13.1   5.84 7.72e-07 5.53e-04
ATTTCAAGAT   -13.2  -5.81 1.00e-06 6.17e-04
GTGCGCTGAG   -30.4  39.31 1.11e-06 6.17e-04
CTTGACATAC   -30.4 -39.20 1.87e-06 9.37e-04
```

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

We note that in general, when using tagwise dispersions, the counts are more consistent within groups, as using tagwise dispersions instead of the common dispersion penalises tags which are highly variable within groups. The smaller the value selected for `prior.n`, the more highly variable tags will be penalised, as there is less 'squeezing' of the tagwise dispersions towards the common value. This effect is seen clearly in the table below (compare this with the corresponding table for the analysis using the common dispersion).

```
> detags.tgw <- rownames(topTags(de.tgw)$table)
> d$counts[detags.tgw, ]

             1  2   3    4
AGCTGTTCCC   0  0 119 1011
CTTGGGTTTT   0  0  21   97
TCACCGGTCA 118 75   6    5
GTCATCACCA  35 20   0    0
TACAAAATCG   0  0  14   56
TAAATTGCAA 103 59   3    6
TAATTTTTGC   0  1  37   21
ATTTCAAGAT  35 21   0    1
GTGCGCTGAG   0  0  18   23
CTTGACATAC  18 20   0    0
```

Of course, we can sort the top table differently, as we did earlier.

We see in the output below that 227 genes are significantly differentially expressed (using Zhang et al. [1997]'s cut off of p-values less than 0.01) according to edgeR when using the

tagwise dispersion estimates (eight fewer than when using the common dispersion). Of those tags, 89 are up-regulated in the cancer cells compared with the normal cells and 138 are down-regulated in the cancer cells compared with normal cells. These proportions of up- and down-regulated tags are similar to those found using the common dispersion, but there is a slightly higher proportion of down-regulated tags in those identified as DE using tagwise dispersions.

```
> sum(de.tgw$table$p.value < 0.01)

[1] 227

> toptgw <- topTags(de.tgw, n = sum(de.tgw$table$p.value < 0.01))
> sum(toptgw$table$logFC > 0)

[1] 89

> sum(toptgw$table$logFC < 0)

[1] 138
```

Furthermore, we see below that 80 tags (1.6% of the total number) have a p-value of less than 0.05 after adjusting for multiple testing using the Benjamini and Hochberg [1995] method for controlling the FDR, which is strong evidence for differential expression.

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

   [,1]
-1   49
0  4932
1    31
```

10.5.3 Visualising DGE results

Shown below is the code for producing the default fold-change plot using `plotSmear` with the DE tags as determined using tagwise dispersions highlighted, and the result of this code is shown in Figure 4.

```
> detags.tgw <- rownames(topTags(de.tgw, n = sum(
+                         de.tgw$table$p.value < 0.01) )$table)
> png(file="edgeR_case_study_Zhang_tgw_smearplot.png", height=600, width=600)
> plotSmear(d, de.tags=detags.tgw, main=
+           "FC plot using tagwise dispersions")
> abline(h = c(-2, 2), col = "dodgerblue")
> dev.off()

null device
          1
```

In Figure 4, the 227 tags identified as differentially expressed (i.e. those identified as significant (p-value less than 0.01) by edgeR using the tagwise dispersions) are highlighted in red. We see that the pattern of differential expression using tagwise dispersions that we see in Figure 4 is very similar to that obtained using the common dispersion that we saw in Figure 2.

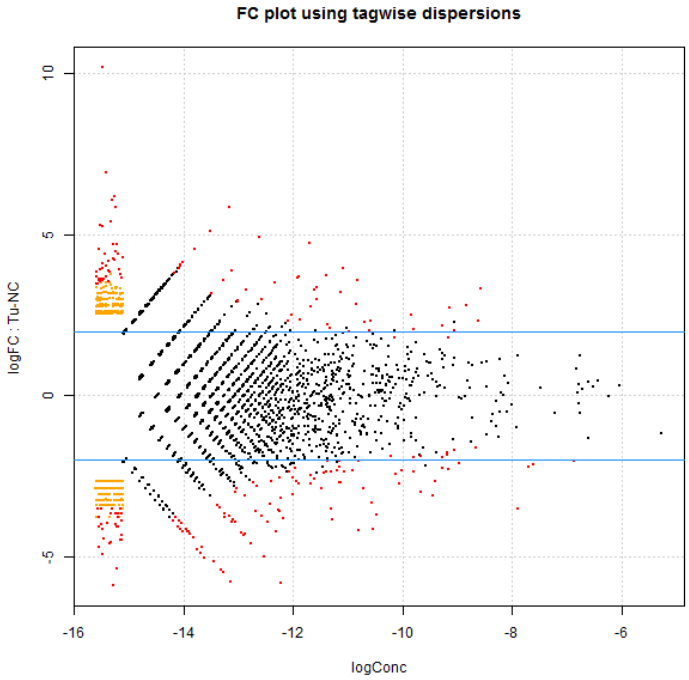

Figure 4: Plot of the log-fold change against the log-concentration for each tag. The 225 most differentially expressed tags as identified by edgeR are outlined in red.

10.6 Setup

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

```
> sessionInfo()

R version 2.14.0 Under development (unstable) (2011-06-15 r56138)
Platform: i386-pc-mingw32/i386 (32-bit)

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

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

other attached packages:
[1] limma_3.9.5  edgeR_2.3.29

loaded via a namespace (and not attached):
[1] tools_2.14.0
```

and took less than one minute to carry out on an Apple MacBook with a 2.8 Ghz Intel Core 2 Duo processor and 8 Gb of 1067 MHz DDR3 memory.

11 Case Study: deep-sequenced short tags

11.1 Introduction

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

11.2 Source of the data

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

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

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

The researchers obtained $\sim$ 2.4 million sequence tags per sample, with tag abundance spanning four orders of magnitude. They found the results to be highly reproducible, even across laboratories. Using a dedicated Bayesian model, they found 3179 transcripts to be differentially expressed with a FDR of 8.5%. This is a much higher figure than was found for

the microarrays. 't Hoen et al. [2008] conclude that deep-sequencing offers a major advance in robustness, comparability and richness of expression profiling data.

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

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

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

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

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

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

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

An object of class "DGEList"
$samples
          files group                          description lib.size
1 GSM272105.txt  DCLK transgenic (Dclk1) mouse hippocampus  2685418
2 GSM272106.txt    WT          wild-type mouse hippocampus  3517977
3 GSM272318.txt  DCLK transgenic (Dclk1) mouse hippocampus  3202246
4 GSM272319.txt    WT          wild-type mouse hippocampus  3558260
5 GSM272320.txt  DCLK transgenic (Dclk1) mouse hippocampus  2460753
6 GSM272321.txt    WT          wild-type mouse hippocampus   294909
```

```
7 GSM272322.txt  DCLK transgenic (Dclk1) mouse hippocampus   651172
8 GSM272323.txt    WT          wild-type mouse hippocampus  3142280
  norm.factors
1            1
2            1
3            1
4            1
5            1
6            1
7            1
8            1

$counts
                  1 2 3 4 5 6 7 8
CATCGCCAGCGGGCACC 1 0 0 0 0 0 0 0
AAGGTCGACTCTGAAGT 1 1 0 0 0 0 0 0
CCTTCCTGGCTCTATGG 1 0 0 0 0 0 0 0
TCTGCTGAGCGTCTGTT 1 0 0 0 0 0 0 0
CCCCAGAGCGAATCAGG 1 1 2 1 1 0 2 1
844311 more rows ...

> colnames(d) <- c("DCLK1","WT1","DCLK2","WT2","DCLK3","WT3","DCLK4","WT4")
```

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

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

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

[1] 44882     8
```

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

11.4 Producing an MDS plot

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

```
> pdf(file="edgeR_case_study_tHoen_MDSplot.pdf", height=6, width=6)
> plotMDS(d, main="MDS Plot for 't Hoen Data", xlim=c(-2,1))

Using grid search to estimate tagwise dispersion.

> dev.off()

null device
          1
```

This function is a variation on the usual multdimensional scaling (or principle coordinate) plot, in that a distance measure particularly appropriate for the digital gene expression (DGE) context is used. The distance between each pair of samples (columns) is the square root of the common dispersion for the top n (default is $n = 500$) genes which best distinguish that pair of samples. These top n genes are selected according to the tagwise dispersion of all the samples. The resulting plot for the 't Hoen data is shown in 5.

11.5 Analysis using common dispersion

11.5.1 Estimating the common dispersion

As discussed for the SAGE data, the first major step in the analysis of DGE data using the NB model is to estimate the dispersion parameter for each tag. Like in the earlier case study, we begin by estimating the common dispersion using the function `estimateCommonDisp`.

```
> system.time( d <- estimateCommonDisp(d) )

   user  system elapsed
  20.36    0.83   21.20

> names(d)

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

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

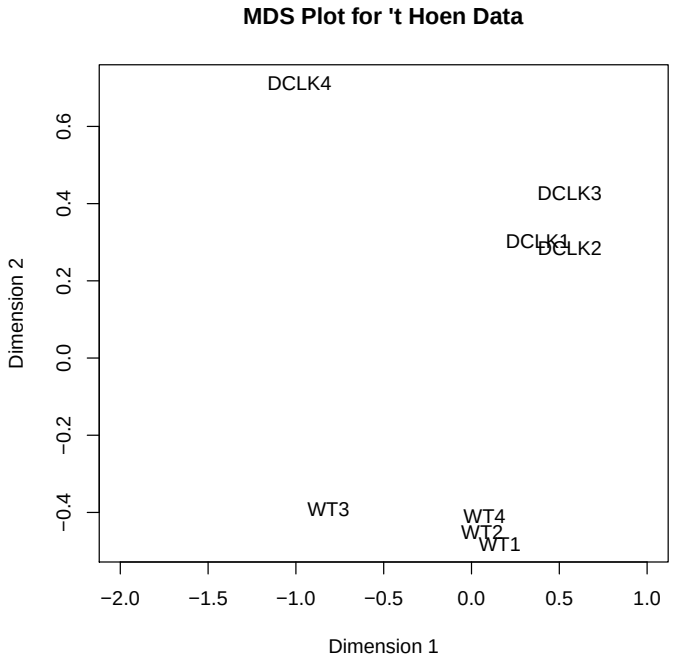

Figure 5: Multidimensional scaling (MDS) plot for the 't Hoen data, showing the relations between the samples in two dimensions. Dimension 1 separates the DCLK and WT samples quite nicely.

```
> d$common.dispersion

[1] 0.151

> sqrt(d$common.dispersion)

[1] 0.389
```

11.5.2 Testing

Once we have an estimate of the common dispersion, we can proceed with testing procedures for determining differential expression. As for the SAGE data, there are only two groups here, so the pair need not be specified in the call to `exactTest`. However, using the `pair` argument allows the user to determine how the comparison is made. Setting `pair=c(''WT'',''DCLK'')` as below means that the comparison done is $DCLK - WT$.

```
> system.time( de.common <- exactTest(d, pair=c("WT","DCLK")) )
```

```
Comparison of groups:  DCLK - WT
   user  system elapsed
     25       0      25
```

The results of the NB exact test can be accessed conveniently using the `topTags` function applied to the object produced by `exactTest`. The table below shows the top 10 DE genes ranked by p-value.

The table in the output from `topTags` shows that the edgeR package identifies a good deal of differential expression between the wild-type and the DCLK-transgenic groups. The top DE tags are given very small p-values, even after adjusting for multiple testing. Furthermore, all of the top tags have a large fold change, indicating that these tags are likely to be biologically meaningful. As suggested in the SAGE case study, a Gene Ontology analysis could be carried out using the list of top tags and p-values provided by `topTags` in order to obtain more systematic and functional information about the differentially expressed genes.

```
> topTags(de.common)

Comparison of groups:  DCLK-WT
                  logConc logFC  P.Value      FDR
CCGTCTTCTGCTTGTCG   -10.6 -5.57 2.02e-30 9.06e-26
CCGTCTTCTGCTTGTAA   -14.4 -5.42 3.36e-27 7.41e-23
TCTGTACGCAGTCAGGC   -18.5  9.73 4.95e-27 7.41e-23
CCGTCTTCTGCTTGTCA   -15.5 -5.46 6.53e-25 7.33e-21
CCGTCTTCTGCTTGTAG   -15.6 -4.75 7.57e-21 6.80e-17
CATAAGTCACAGAGTCG   -32.8 34.51 2.27e-15 1.69e-11
TCTGTATGTTCTCGTAT   -16.1 -4.11 4.44e-15 2.85e-11
CCGTCTTCTGCTTGAAA   -12.0 -3.36 2.65e-14 1.49e-10
ATACTGACATTTCGTAT   -16.8 -4.16 3.86e-14 1.93e-10
AAAAGAAATCACAGTTG   -33.0 34.11 1.14e-13 5.12e-10
```

The table below shows the raw counts for the tags that edgeR has identified as the most differentially expressed. For these tags there seem to be very large differences between the groups, suggesting that the DE genes identified are truly differentially expressed, and not false positives. We do see, however, that when a common dispersion value is used, tags which have just one large count (counts per million shown below) in one sample can appear as highly DE. If we wish to give less significance to such tags then we can use tagwise dispersion estimates as described below.

```
> detags.com <- rownames(topTags(de.common)$table)
> cpm.d[detags.com, order(d$samples$group)]

                   DCLK1 DCLK2  DCLK3 DCLK4   WT1     WT2     WT3   WT4
CCGTCTTCTGCTTGTCG  39.472 83.69 244.23  7.68 422.1 118.035 17483.4 77.01
CCGTCTTCTGCTTGTAA   4.469  6.56  12.60  1.54  24.7   7.869  1193.6  4.46
TCTGTACGCAGTCAGGC  59.581 31.54 178.81 50.68   0.0   0.281     0.0  0.00
CCGTCTTCTGCTTGTCA   0.745  2.50   7.72  1.54  11.9   4.778   620.5  5.41
```

```
CCGTCTTCTGCTTGTAG   3.351  3.44   6.91  0.00  17.3   5.621   451.0  2.86
CATAAGTCACAGAGTCG  24.950 24.05  23.57 10.75   0.0   0.000     0.0  0.00
TCTGTATGTTCTCGTAT   2.979  3.75   2.84  4.61  26.7 120.002    20.3 56.33
CCGTCTTCTGCTTGAAA 100.171 76.20  53.24 70.64  28.7  16.019  3095.9 13.68
ATACTGACATTTCGTAT   1.862  1.56   3.25  1.54  32.1  64.076    13.6 33.10
AAAAGAAATCACAGTTG  11.544 28.11  17.07  4.61   0.0   0.000     0.0  0.00
```

If we order the tags by fold change instead of p-value, as in the table below, we see that the genes with the largest fold changes have very small concentrations, and in general the p-values are not as small as when ranked by p-value (not surprisingly). This ranking is dominated by genes that have zero total counts in one group and is less informative than ranking by p-value.

```
> topTags(de.common, sort.by = "logFC")

Comparison of groups:  DCLK-WT
                  logConc logFC  P.Value      FDR
CATAAGTCACAGAGTCG   -32.8  34.5 2.27e-15 1.69e-11
AAAAGAAATCACAGTTG   -33.0  34.1 1.14e-13 5.12e-10
CAAACTAGAAGACAGAA   -33.3  33.4 1.47e-10 2.27e-07
TCTCTAAAGCGTCCTTG   -33.6  32.8 3.58e-08 2.68e-05
GAAGGAGTCTTCGTATG   -34.0 -32.1 4.37e-06 1.32e-03
TTAATATCTTTCGTATG   -34.0 -32.0 9.21e-06 2.19e-03
AATAAATATCTTTTTCT   -34.2 -31.6 7.22e-06 1.88e-03
GTGCTGGAGGAGACAAG   -34.3  31.5 1.48e-04 1.27e-02
CATACAGTGCCAGTCGT   -34.3  31.5 3.44e-04 2.06e-02
AGTCGTATGCCGTCTTC   -34.5  31.1 8.22e-04 3.41e-02
```

Using their dedicated Bayesian model, 't Hoen et al. [2008] found 3179 transcripts to be differentially expressed with a FDR of 8.5%. We can compare 't Hoen et al. [2008]'s results with the results from edgeR by applying the `topTags` function to help look at the tags that have a FDR of less than 0.085 after adjusting for multiple testing using Benjamini and Hochberg [1995]'s method for controlling the FDR.

We see in the output below that 2286 tags (5.1% of the total number analysed) are significantly differentially expressed according to edgeR using the common dispersion estimate. Of those tags, 753 (33% of the DE tags) are up-regulated in the wild-type compared with the transgenic samples and 1533 (67%) are down-regulated in the wild-type compared with transgenic mice.

```
> summary(decideTestsDGE(de.common, p.value=0.085))

   [,1]
-1   753
0  42596
1   1533
```

11.5.3 Visualising DGE results

The code for producing the default fold-change plot, with the top 500 most DE tags highlighted in red, is shown below, and the result of this code is shown in Figure 6. In Figure 6, we see that the 500 tags identified as most differentially expressed have large fold changes—almost all of the 500 tags in red fall outside the blue lines at $\log \mathrm{FC} = -2$ and $\log \mathrm{FC} = 2$. This means that most of these tags show at least a 4-fold change in expression level between the samples. This plot suggests strongly that the tags identified by edgeR as differentially expressed are truly differentially expressed, and, given the large changes in expression level, are likely to be biologically meaningful.

```
> detags500.com <- rownames(topTags(de.common, n = 500)$table)
> png(file="edgeR_case_study_tHoen_smearplot.png", height=600, width=600)
> plotSmear(de.common, de.tags = detags500.com, main = "FC plot using common dispersion", cex=0.6)
> abline(h = c(-2, 2), col = "dodgerblue", lwd = 2)
> dev.off()

null device
          1
```

11.6 Analysis using moderated tagwise dispersions

11.6.1 Moderating the tagwise dispersion

As discussed in the previous case studies, an extension to simply using the common dispersion for each tag is to estimate the dispersion separately for each tag, while 'squeezing' these estimates towards the common dispersion estimate. The goal of this moderation of the dispersion estimates is to improve inference by sharing information between tags. This type of analysis can be carried out in few steps using the edgeR package.

To run the moderated analysis, we need to determine how much moderation is necessary. As discussed above, we currently prefer to choose a sensible value for the smoothing parameter *a priori*, although we do have an algorithm developed by Robinson and Smyth [2007] for estimating the smoothing parameter as an approximate eBayes rule.

In our experience analysing RNA-Seq data we have found that a good rule of thumb for choosing a value for `prior.n` is to choose a certain number of prior degrees of freedom (a value between 20 and 30 works well) and then divide this number by the degrees of freedom (number of samples minus the number of variables being fit in the model; in the case of a simple multiple-group comparison this is simply the number of samples minus the number of groups). The function `getPriorN` automatically calculates the appropriate value for `prior.n` for a given experimental design.

As we only have eight libraries, a small sample size, we should not be too confident about the accuracy of the tagwise dispersions. In this experiment we have eight samples and two groups, which means we have six degrees of freedom for estimating the dispersion parameter.

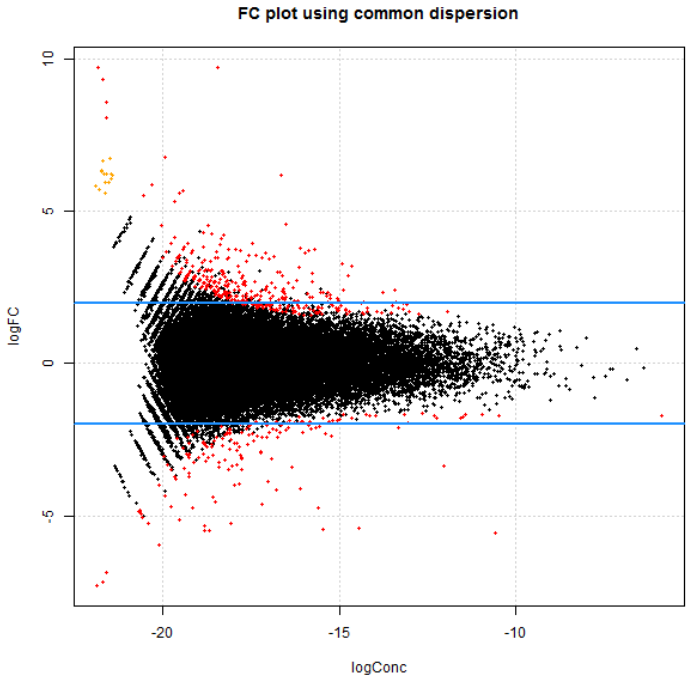

Figure 6: Plot of the log-fold change against the log-concentration for each tag. The 500 most differentially expressed tags as identified by edgeR using the common dispersion are outlined in red.

Thus, setting the `prior.n` to be 3.33 (20 divided by six) should be appropriate. This means that the common likelihood receives the weight of 3.33 individual tags. Therefore, there will be a reasonable degree of 'squeezing' towards the common dispersion estimate, but still enough scope to allow flexibility when estimating the individual dispersion for each gene. By default, `estimateTagwiseDisp` uses the `prior.n` value from `getPriorN`.

The function `estimateTagwiseDisp` produces a DGEList object that contains all of the elements present in the object produced by `estimateCommonDisp`, as well as the value for `prior.n` used (`d$prior.n`) and the tagwise dispersion estimates (`d$tagwise.dispersion`), as we see below. Here we set `grid.length=500` for greater precision in the tagwise dispersion estimates. We use the `trend` argument below to indicate that we wish to allow a dependence of the dispersion on the overall expression level.

```
> system.time( d <- estimateTagwiseDisp(d, prop.used=0.5,
+                              trend=TRUE, grid.length=500) )

Using grid search to estimate tagwise dispersion.
   user  system elapsed
  66.41    0.78   67.32
```

```
> names(d)

 [1] "samples"            "common.dispersion"  "prior.n"
 [4] "tagwise.dispersion" "counts"             "pseudo.alt"
 [7] "genes"              "all.zeros"          "conc"
[10] "common.lib.size"

> d$prior.n

[1] 3.33

> head(d$tagwise.dispersion)

[1] 0.153 0.200 0.122 0.239 0.127 0.170
```

It is interesting to consider the distribution of the tagwise dispersion estimates. As we can see from the output below, the tagwise dispersion estimates range from a minimum of 0.08 to a maximum of 1.05, and the common dispersion estimate lies in between the median and mean values for the tagwise dispersion estimates. Figure 7 shows the relationship between the estimated tagwise dispersions and tag abundance (log-concentration) for this dataset.

```
> summary(d$tagwise.dispersion)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.081   0.133   0.170   0.180   0.209   1.050

> d$common.dispersion

[1] 0.151

> png(file="edgeR_case_study_tHoen_tgw_disp_vs_logconc.png", height=600, width=600)
> plot(log(d$conc$conc.common), d$tagwise.dispersion, panel.first=grid(),
+      ylab="tagwise dispersion", xlab="logConc")
> abline(h=d$common.dispersion, col="dodgerblue", lwd=3)
> dev.off()

null device
          1
```

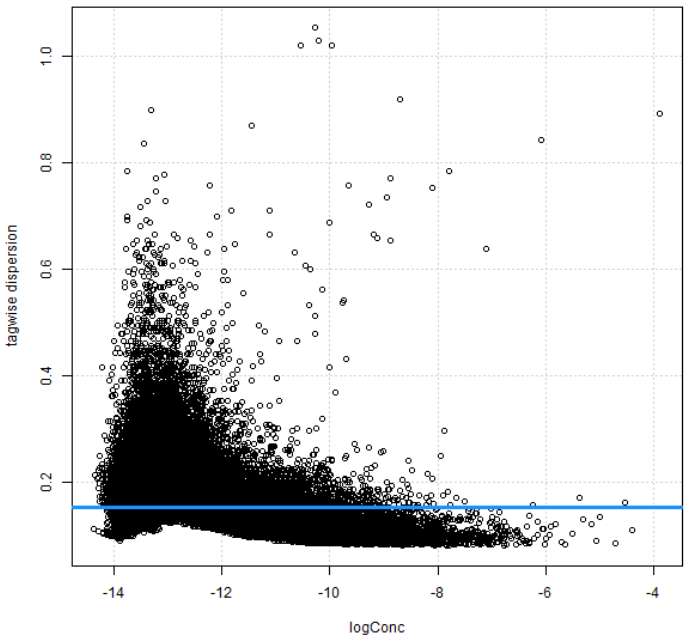

Figure 7: Plot of the tagwise dispersions against tag abundance (log-concentration).

11.6.2 Testing

Once we have an estimate of the common dispersion and/or estimates of the tagwise dispersions, we can proceed with testing procedures for determining differential expression using `exactTest`. Here we carry out the testing using the tagwise dispersion estimates calculated using a `prior.n` value of ten.

By default, `exactTest` uses the common dispersion, but by adding the argument `common.disp=FALSE`, tagwise dispersion estimates will be used instead.

```
> de.tagwise <- exactTest(d, common.disp = FALSE, pair=c("WT","DCLK"))

Comparison of groups:  DCLK - WT
```

Just as we saw earlier, the object produced by `exactTest` contains two elements. The first is a data frame (`table`) that contains the elements `logConc`, `logFC` and `p.value` and the second is a vector (`comparison`) that lists the names of the groups being compared.

The output below shows that when using tagwise dispersions, the edgeR package still identifies a lot of differential expression between the wild-type group and the DCLK-transgenic group. The top DE tags are given very small p-values, even after adjusting for multiple

testing. However, We see immediately that the p-values for the top tags are many orders of magnitude greater than those for the top tags identified using the common dispersion.

As with the analysis using the common dispersion, all of the top tags have a large fold change, indicating that these changes in expression are likely to be biologically meaningful, although interestingly we see more tags (7 out of 10) that are down-regulated in the wild-type group compared with the DCLK group, which contrasts with using the common dispersion. We note that the ranking of the tags is different, too, and only three of the top ten tags according to using the common dispersion are found to be among the top ten tags using tagwise dispersions.

```
> topTags(de.tagwise)

Comparison of groups:  DCLK-WT
                  logConc logFC  P.Value      FDR
TCTGTACGCAGTCAGGC   -18.5  9.73 7.18e-23 3.22e-18
CATAAGTCACAGAGTCG   -32.8 34.52 7.36e-17 1.65e-12
GCTAATAAATGGCAGAT   -14.9  3.28 2.61e-15 3.91e-11
ATACTGACATTTCGTAT   -16.8 -4.16 2.35e-14 2.34e-10
CCAAGAATCTGGTCGTA   -17.5  3.93 2.61e-14 2.34e-10
TTCCTGAAAATGTGAAG   -17.1  3.64 1.50e-13 1.04e-09
CTGCTAAGCAGAAGCAA   -17.0  3.42 1.62e-13 1.04e-09
TCTGTATGTTCTCGTAT   -16.1 -4.10 3.38e-13 1.90e-09
CCTATTTTTCTCTCGTA   -14.6  3.19 4.88e-13 2.43e-09
TATTTTGTTTTGTCGTA   -17.0 -3.88 5.58e-13 2.51e-09
```

The tables below shows the raw counts for the genes that edgeR has identified as the most differentially expressed, using the common dispersion and tagwise dispersions. For these tags, using both methods, there seem to be very large differences between the groups, suggesting that the DE genes identified are truly differentially expressed, and not false positives.

Particularly noteworthy, however, is how much more consistent the counts *within* groups are for the top tags identified using tagwise dispersions compared with those identified using the common dispersion. This is to be expected, as allowing tagwise dispersions penalises highly variable tags, so those that have greater variability within groups (especially one or two libraries with extremely high counts) will appear far lower in the ranking using tagwise dispersions than they would using the common dispersion. This difference in the rankings provided by the two approaches to the dispersion parameter could yield valuable information. The counts per million are shown for the top 10 DE tags according to the tagwise and common dispersion approaches. The tagwise dispersion estimates are shown for the to tags according to the tagwise approach. We see that when we use the tagwise dispersions the DE tags have more consistent expression levels within groups.

```
> detags.tgw <- rownames(topTags(de.tagwise)$table)
> detags.com <- rownames(topTags(de.common)$table)
> tgw.disp <- d$tagwise.dispersion
> names(tgw.disp) <- rownames(d)
> cbind(cpm.d[detags.tgw, order(d$samples$group)], tgw.disp[detags.tgw] )
```

```
                    DCLK1  DCLK2  DCLK3  DCLK4    WT1      WT2   WT3   WT4
TCTGTACGCAGTCAGGC  59.58  31.54 178.81  50.68  0.000    0.281  0.00  0.00 0.198
CATAAGTCACAGAGTCG  24.95  24.05  23.57  10.75  0.000    0.000  0.00  0.00 0.127
GCTAATAAATGGCAGAT 144.11 100.24  53.64 109.03 12.791    8.993  3.39 12.09 0.122
ATACTGACATTTCGTAT   1.86   1.56   3.25   1.54 32.121   64.076 13.56 33.10 0.148
CCAAGAATCTGGTCGTA  26.07  20.61  19.10  19.96  0.853    1.405  0.00  2.23 0.112
TTCCTGAAAATGTGAAG  27.56  21.86  34.95  15.36  1.706    2.529  0.00  2.23 0.122
CTGCTAAGCAGAAGCAA  28.30  27.48  21.13  23.04  1.990    1.967  0.00  3.50 0.110
TCTGTATGTTCTCGTAT   2.98   3.75   2.84   4.61 26.720  120.002 20.35 56.33 0.181
CCTATTTTTCTCTCGTA  83.79 144.59 191.81  50.68 15.634   15.457  3.39 11.77 0.148
TATTTTGTTTTGTCGTA   3.72   1.56   1.22   0.00 25.014   48.057 13.56 21.32 0.153

> cpm.d[detags.com, order(d$samples$group)]

                     DCLK1 DCLK2  DCLK3 DCLK4   WT1      WT2      WT3   WT4
CCGTCTTCTGCTTGTCG  39.472 83.69 244.23  7.68 422.1 118.035 17483.4 77.01
CCGTCTTCTGCTTGTAA   4.469  6.56  12.60  1.54  24.7   7.869  1193.6  4.46
TCTGTACGCAGTCAGGC  59.581 31.54 178.81 50.68   0.0   0.281     0.0  0.00
CCGTCTTCTGCTTGTCA   0.745  2.50   7.72  1.54  11.9   4.778   620.5  5.41
CCGTCTTCTGCTTGTAG   3.351  3.44   6.91  0.00  17.3   5.621   451.0  2.86
CATAAGTCACAGAGTCG  24.950 24.05  23.57 10.75   0.0   0.000     0.0  0.00
TCTGTATGTTCTCGTAT   2.979  3.75   2.84  4.61  26.7 120.002    20.3 56.33
CCGTCTTCTGCTTGAAA 100.171 76.20  53.24 70.64  28.7  16.019  3095.9 13.68
ATACTGACATTTCGTAT   1.862  1.56   3.25  1.54  32.1  64.076    13.6 33.10
AAAAGAAATCACAGTTG  11.544 28.11  17.07  4.61   0.0   0.000     0.0  0.00

> topTags(de.common)

Comparison of groups:  DCLK-WT
                   logConc logFC  P.Value      FDR
CCGTCTTCTGCTTGTCG    -10.6 -5.57 2.02e-30 9.06e-26
CCGTCTTCTGCTTGTAA    -14.4 -5.42 3.36e-27 7.41e-23
TCTGTACGCAGTCAGGC    -18.5  9.73 4.95e-27 7.41e-23
CCGTCTTCTGCTTGTCA    -15.5 -5.46 6.53e-25 7.33e-21
CCGTCTTCTGCTTGTAG    -15.6 -4.75 7.57e-21 6.80e-17
CATAAGTCACAGAGTCG    -32.8 34.51 2.27e-15 1.69e-11
TCTGTATGTTCTCGTAT    -16.1 -4.11 4.44e-15 2.85e-11
CCGTCTTCTGCTTGAAA    -12.0 -3.36 2.65e-14 1.49e-10
ATACTGACATTTCGTAT    -16.8 -4.16 3.86e-14 1.93e-10
AAAAGAAATCACAGTTG    -33.0 34.11 1.14e-13 5.12e-10
```

We might also be interested in comparing the top-ranking genes as identified by edgeR using the common dispersion and tagwise dispersions. The output below shows, firstly, that there are four tags that appear in the top ten most DE tags using both common and tagwise dispersions. Secondly, we see that of the top 1000 most DE tags as identified using tagwise dispersions, 77% of these tags are also in the list of the 1000 most DE tags as identified using the common dispersion. This shows that although we do get quite different results depending on which method we use, there is still a great deal of agreement as to which tags are DE.

```
> sum(rownames(topTags(de.tagwise)$table) %in%
+     rownames(topTags(de.common)$table))

[1] 4

> sum(rownames(topTags(de.tagwise, n = 1000)$table) %in%
+     rownames(topTags(de.common, n = 1000)$table))/1000 * 100

[1] 76.5
```

Using their dedicated Bayesian model, 't Hoen et al. [2008] found 3179 transcripts to be differentially expressed with a FDR of 8.5%. The output below shows that using Benjamini and Hochberg [1995]'s approach for controlling the FDR at 8.5%, edgeR identifies 2286 tags as DE using common dispersion and 2209 tags as DE using tagwise dispersions. This means that we determine 5.1% and 4.9% of tags to be DE using common and tagwise dispersions, respectively. The `decideTestsDGE` function provides a useful way to summarize DE results after testing, as shown below.

```
> summary(decideTestsDGE(de.common, p.value=0.085))

   [,1]
-1   753
0  42596
1   1533

> mean(p.adjust(de.common$table$p.value,
+               method = "BH") < 0.085) * 100

[1] 5.09

> summary(decideTestsDGE(de.tagwise, p.value=0.085))

   [,1]
-1   722
0  42673
1   1487

> mean(p.adjust(de.tagwise$table$p.value,
+               method = "BH") < 0.085) * 100

[1] 4.92

> summary(decideTestsDGE(de.tagwise, p=0.085))

   [,1]
-1   722
0  42673
1   1487
```

Of the 2209 tags identified as DE using tagwise dispersions, 722 (33%) are up-regulated in wild-type and 1487 (67%) are up-regulated in the transgenic mice. The proportions of up- and down-regulated genes identified using the two approaches to modeling the dispersion are similar, but using the common dispersion identifies slightly more tags down-regulated in wild-type mice as DE.

11.6.3 Visualising DGE results

As discussed earlier, the function `plotSmear` can be used to generate a plot of the log-fold change against the log-concentration for each tag (analogous to an MA-plot in the microarray context). We identify the top 500 most DE tags using both common dispersion and tagwise dispersions so we can highlight them on the plots and compare what we see. The code for producing the fold-change plots is shown below, and the result of this code is shown in Figure 8.

```
> detags500.com <- rownames(topTags(de.common, n = 500)$table)
> detags500.tgw <- rownames(topTags(de.tagwise, n = 500)$table)
> png(file="edgeR_case_study_tHoen_comparative_smearplots.png", height=800, width=600)
> par(mfcol = c(2, 1))
> plotSmear(de.common, de.tags = detags500.com, main = "Using common dispersion")
> abline(h = c(-2, 2), col = "dodgerblue", lwd = 2)
> plotSmear(de.tagwise, de.tags = detags500.tgw, main = "Using tagwise dispersions")
> abline(h = c(-2, 2), col = "dodgerblue", lwd = 2)
> dev.off()

null device
          1
```

In Figure 8, the top 500 most differentially expressed tags (those identified as significant by edgeR using the common dispersion (top) and tagwise dispersions (bottom)) are highlighted in red. Looking at Figure 8, we see that, generally speaking, the pattern of differential expression looks similar using the two different methods, and the tags identified as DE have convincingly large fold changes.

11.7 Setup

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

```
> sessionInfo()

R version 2.14.0 Under development (unstable) (2011-06-15 r56138)
Platform: i386-pc-mingw32/i386 (32-bit)

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

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

other attached packages:
[1] limma_3.9.5  edgeR_2.3.29
```

```
loaded via a namespace (and not attached):
[1] tools_2.14.0
```

and took less than 5 minutes to carry out on an Apple MacBook with a 2.8 Ghz Intel Core 2 Duo processor and 8 Gb of 1067 MHz DDR3 memory.

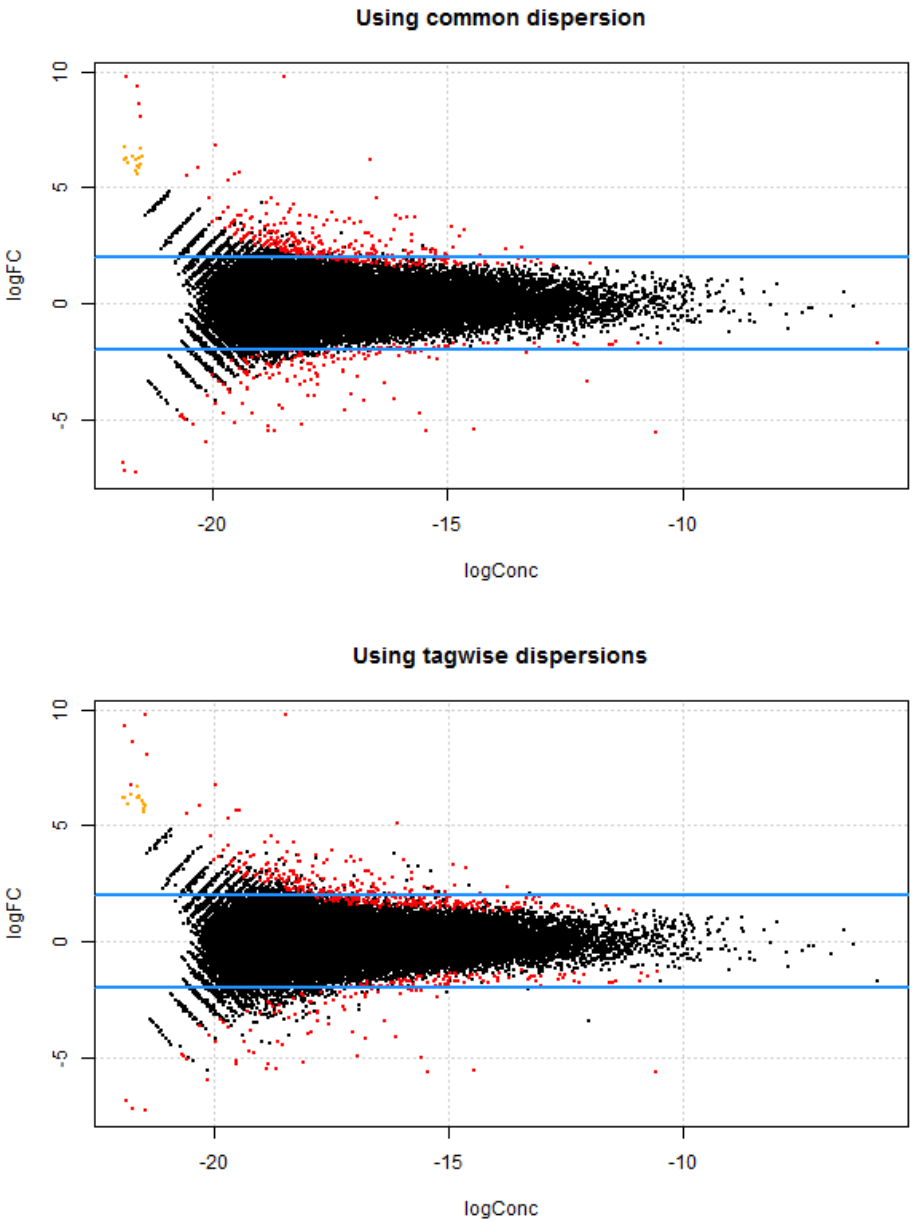

Figure 8: Plots of the log-fold change against the log-concentration for each tag, using the common dispersion (upper), and tagwise dispersions (lower). Tags with positive fold-change here are up-regulated in wild-type compared with transgenic mice. The 500 most differentially expressed tags according to each method are highlighted in red on both plots.

12 Case Study: RNA-seq data

12.1 Introduction

This section provides a detailed analysis of data from a study by Li et al. [2008] designed to address a range of practical issues in RNA-seq experiments:

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

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

12.2 Source of the data

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

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

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

12.3 Reading in the data and creating a DGEList object

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

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

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

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

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

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

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

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

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

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

```
[1] 37435     7

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

An object of class "DGEList"
$samples
        group lib.size norm.factors
lane1 Control   978576        1.030
lane2 Control  1156844        1.037
lane3 Control  1442169        1.036
lane4 Control  1485604        1.038
lane5     DHT  1823460        0.954
lane6     DHT  1834335        0.953
lane8     DHT   681743        0.958

$counts
                lane1 lane2 lane3 lane4 lane5 lane6 lane8
ENSG00000124208   478   619   628   744   483   716   240
ENSG00000182463    27    20    27    26    48    55    24
ENSG00000124201   180   218   293   275   373   301    88
ENSG00000124207    76    80    85    97    80    81    37
ENSG00000125835   132   200   200   228   280   204    52
16489 more rows ...

$all.zeros
ENSG00000124208 ENSG00000182463 ENSG00000124201 ENSG00000124207 ENSG00000125835
          FALSE           FALSE           FALSE           FALSE           FALSE
16489 more elements ...
```

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

12.4 Producing an MDS plot

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

```
> pdf(file="edgeR_case_study_Li_MDSplot.pdf", height=6, width=6)
> plotMDS(d, main="MDS Plot for Li Data", xlim=c(-1,1), labels=c(
+       "Control1","Control2","Control3","Control4","DHT1","DHT2","DHT3"))
```

```
Using grid search to estimate tagwise dispersion.

> dev.off()

null device
          1
```

The resulting plot for the Li data is shown in 9. In this plot, Dimension 1 clearly separates the Control from the DHT-treated samples. This shows that the replicates are reasonably similar to each other and that we can expect to find lots of DE genes. Having now investigated some of the relationships between the samples we can proceed to the DE analysis of the data.

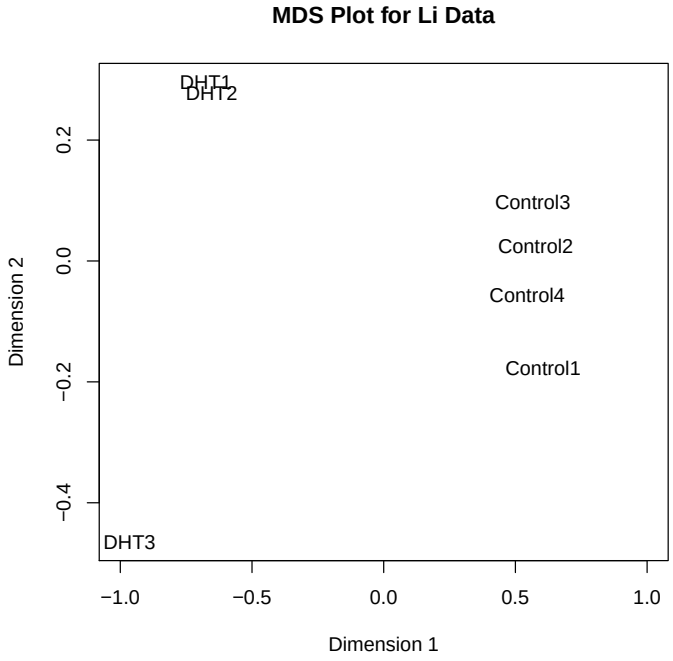

Figure 9: Multidimensional scaling (MDS) plot for the Li data, showing the degree of similarity between the samples in two dimensions. We see that Dimension 1 strongly separates the Control from the DHT-treated samples. There are no outliers on this plot.

12.5 Analysis using common dispersion

12.5.1 Estimating the common dispersion

As discussed for the SAGE data, the first major step in the analysis of DGE data using the NB model is to estimate the dispersion parameter for each tag. Like in the earlier case study,

we begin by estimating the common dispersion using the function `estimateCommonDisp`, and analysing the data using the common dispersion.

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

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

The output of `estimateCommonDisp` is a DGEList object with several new elements. The element `common.dispersion`, as the name suggests, provides the estimate of the common dispersion. The pseudocounts calculated under the alternative hypothesis are given by `pseudo.alt`. The element `conc` gives the estimates of the overall concentration of each tag across all of the original samples (`conc$conc.common`) and the estimate of the concentration of each tag within each group (`conc$conc.group`). The element `common.lib.size` gives the library size to which the original libraries have been adjusted in the pseudocounts.

Here the coefficient of variation of biological variation (square root of the common dispersion) is found to be 0.142. We also note that although a common dispersion estimate of 0.02 may seem 'small', if a tag has just an average of just 200 counts per sample, then the estimate of the tag's variance is 5 times greater than it would be under the Poisson model.

```
> d$common.dispersion

[1] 0.02

> sqrt(d$common.dispersion)

[1] 0.141
```

12.5.2 Testing

Once we have an estimate of the common dispersion, we can proceed with testing procedures for determining differential expression. As for the SAGE data, there are only two groups here, so the pair need not be specified in the call to `exactTest`.

```
> de.com <- exactTest(d)

Comparison of groups:  DHT - Control

> names(de.com)

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

The results of the NB exact test can be accessed conveniently using the `topTags` function applied to the object produced by `exactTest`. The table below shows the top 10 DE genes ranked by p-value. The output of `topTags` displays the log-concentration (measure of overall gene expression), log-fold change (change in expression between groups being compared, computed from the concentration of the gene in each group), the p-value from the exact test and the p-value after adjusting for multiple testing (default is to return the false discovery rate). The argument `n` can be altered to show a different number of "top" genes.

The table in the output from `topTags` shows that the edgeR package identifies a great deal of differential expression, and gives the top genes extremely small p-values, even after adjusting for multiple testing. Furthermore, all of the top genes have a very large fold change (indicating that these tags are likely to be biologically meaningful), and all are up-regulated in the DHT-treatment group compared to the control group.

Of course, for many applications the ranking for differential expression is more important than the p-value, and `topTags` provides such a ranking. As suggested in the SAGE case study, a Gene Ontology analysis could be carried out using the list of top gene and p-values provided by `topTags` in order to obtain more systematic and functional information about the differentially expressed genes.

```
> topTags(de.com)

Comparison of groups:  DHT-Control
                logConc logFC   P.Value       FDR
ENSG00000151503   -11.9  5.82 6.51e-193 1.07e-188
ENSG00000096060   -11.3  5.01 1.23e-162 1.02e-158
ENSG00000127954   -15.6  8.24 2.43e-153 1.34e-149
ENSG00000166451   -12.3  4.69 1.05e-134 4.32e-131
ENSG00000131016   -14.4  5.31 4.12e-110 1.36e-106
ENSG00000113594   -12.8  4.12 5.31e-102  1.46e-98
ENSG00000116285   -13.6  4.21  4.63e-93  1.09e-89
ENSG00000123983   -12.1  3.66  1.19e-92  2.46e-89
ENSG00000166086   -15.2  5.51  1.91e-90  3.51e-87
ENSG00000162772   -10.8  3.32  7.15e-86  1.18e-82
```

The table below shows the raw counts for the genes that edgeR has identified as the most differentially expressed. For these genes there seems to be very large differences between the groups, suggesting that the DE genes identified are truly differentially expressed.

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

                lane1  lane2  lane3  lane4 lane5 lane6 lane8
ENSG00000151503  35.8  30.25  33.98  39.71  1814  1875  1795
ENSG00000096060  66.4  68.29  72.81  76.06  2180  2032  2128
ENSG00000127954   0.0   0.00   2.08   2.02   333   328   323
ENSG00000166451  41.9  44.95  39.52  38.37   960   902  1068
ENSG00000131016   9.2   4.32  12.48   4.04   309   206   312
```

```
ENSG00000113594  37.8  31.12  39.52  28.94   513   523   613
ENSG00000116285  18.4  24.20  15.95  21.54   354   343   320
ENSG00000123983  63.4  65.70  65.18  72.70   743   686   921
ENSG00000166086   9.2   1.73   2.08   4.04   162   162   177
ENSG00000162772 175.8 176.34 173.35 204.63  1630  1782  1631
```

If we order the genes by fold change instead of p-value, we see that the genes with the largest fold changes have very small concentrations. This ranking is dominated by genes that have zero total counts in one group and is less informative than ranking by p-value.

```
> topTags(de.com, n = 10, sort.by = "logFC")

Comparison of groups:  DHT-Control
                logConc logFC  P.Value      FDR
ENSG00000091972   -31.8 -36.5 1.24e-54 5.53e-52
ENSG00000164120   -32.2  35.5 3.12e-45 1.05e-42
ENSG00000100373   -33.0 -34.1 1.07e-16 5.73e-15
ENSG00000118513   -33.0 -34.0 3.35e-15 1.55e-13
ENSG00000081237   -33.2 -33.7 1.41e-12 4.65e-11
ENSG00000196660   -33.2 -33.5 1.70e-11 4.87e-10
ENSG00000117245   -33.3 -33.5 2.85e-11 7.93e-10
ENSG00000019549   -33.4  33.3 2.36e-13 8.39e-12
ENSG00000059804   -33.4  33.2 1.02e-12 3.40e-11
ENSG00000018625   -33.4  33.2 2.13e-12 6.81e-11
```

We can see how many genes are identified as differentially expressed between the control group (untreated LNCaP cells) and the DHT-treated LNCaP cells, for a given threshold for the exact p-value or for the adjusted p-value.

As the output below shows, edgeR detects a huge number of differentially expressed genes in this dataset. Over 3000 genes are given an adjusted p-value less than 0.01.

```
> summary(decideTestsDGE(de.com, p.value=0.01))

   [,1]
-1  1620
0  13110
1   1764
```

The output below shows that 4936 genes are given an adjusted p-value of less than 0.05. This means that if we set out to control the FDR for differential expression at 5%, then edgeR identifies 30% of all the genes in the dataset as differentially expressed.

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

   [,1]
-1  2463
0  11558
1   2473
```

Of the genes identified as DE above, 2463 (49.9% of the DE genes) are up-regulated in DHT-treated compared with control cells, and 2473 (50.1%) are up-regulated in the control cells compared with DHT-treated cells. It is interesting to note that although we detect a handful more genes as DE that are up-regulated in the control group, all of the top ten genes were up-regulated in the DHT-treated group.

12.5.3 Visualising DGE results

The code for producing the default fold-change plot, with the top 500 most DE tags highlighted in red, is shown below, and the result of this code is shown in Figure 10. In Figure 10, we see that the 500 tags identified as most differentially expressed have large fold changes—almost all of the 500 tags in red fall outside the blue lines at $\log \mathrm{FC} = -2$ and $\log \mathrm{FC} = 2$. This means that most of these tags show at least a 4-fold change in expression level between the samples. This plot suggests strongly that the tags identified by edgeR as differentially expressed are truly differentially expressed, and, given the large changes in expression level, are likely to be biologically meaningful.

```
> detags500.com <- rownames(topTags(de.com, n = 500)$table)
> png(file="edgeR_case_study_Li_smearplot.png", height=600, width=600)
> plotSmear(de.com, de.tags = detags500.com, main = "FC plot using common dispersion")
> abline(h = c(-2, 2), col = "dodgerblue", lwd = 2)
> dev.off()

null device
          1
```

12.6 Analysis using moderated tagwise dispersions

12.6.1 Moderating the tagwise dispersion

As discussed in the previous case studies, an extension to simply using the common dispersion for each tag is to estimate the dispersion separately for each tag, while 'squeezing' these estimates towards the common dispersion estimate. The goal of this moderation of the dispersion estimates is to improve inference by sharing information between tags. This type of analysis can be carried out in few steps using the edgeR package.

To run the moderated analysis, we need to determine how much moderation is necessary. As discussed above, we currently prefer to choose a sensible value for the smoothing parameter *a priori*, although we do have an algorithm developed by Robinson and Smyth [2007] for estimating the smoothing parameter as an approximate eBayes rule.

In our experience analysing RNA-Seq data we have found that a good rule of thumb for choosing a value for `prior.n` is to choose a certain number of prior degrees of freedom (a value between 20 and 30 works well) and then divide this number by the degrees of freedom (number of samples minus the number of variables being fit in the model; in the case of a

Figure 10: Plot of the log-fold change against the log-concentration for each tag. The 500 most differentially expressed tags as identified by edgeR using the common dispersion are outlined in red.

simple multiple-group comparison this is simply the number of samples minus the number of groups). The function `getPriorN` automatically calculates the appropriate value for `prior.n` for a given experimental design.

As we only have seven libraries, a small sample size, we should not be too confident about the accuracy of the tagwise dispersions. In this experiment we have seven samples and two groups, which means we have five degrees of freedom for estimating the dispersion parameter. Thus, setting the `prior.n` to be four (20 divided by five) should be appropriate. This means that the common likelihood receives the weight of four individual tags. Therefore, there will be a reasonable degree of 'squeezing' towards the common dispersion estimate, but still enough scope to allow flexibility when estimating the individual dispersion for each gene. By default, `estimateTagwiseDisp` uses the `prior.n` value from `getPriorN`.

The function `estimateTagwiseDisp` produces a DGEList object that contains all of the elements present in the object produced by `estimateCommonDisp`, as well as the value for `prior.n` used (`d$prior.n`) and the tagwise dispersion estimates (`d$tagwise.dispersion`), as we see below. Here we set `grid.length=500` for greater precision in the tagwise dispersion estimates. We use the `trend` argument below to indicate that we wish to allow a dependence

of the dispersion on the overall expression level.

```
> getPriorN(d)

[1] 4

> system.time(
+ d <- estimateTagwiseDisp(d, trend=TRUE, prop.used=0.5, grid.length = 500)
+             )

Using grid search to estimate tagwise dispersion.
   user  system elapsed
  16.89    0.29   17.21

> names(d)

 [1] "samples"            "common.dispersion"  "prior.n"
 [4] "tagwise.dispersion" "counts"             "pseudo.alt"
 [7] "genes"              "all.zeros"          "conc"
[10] "common.lib.size"

> d$prior.n

[1] 4

> head(d$tagwise.dispersion)

[1] 0.0152 0.0235 0.0152 0.0132 0.0215 0.0173
```

It is interesting to consider the distribution of the tagwise dispersion estimates. As we can see from the output below, the tagwise dispersion estimates range from a minimum of 0.007 to a maximum of 0.8, and most of the tagwise dispersion estimates are greater than the common dispersion estimate. Here we have also allowed for a mean-dependent trend on the tagwise dispersion values, which can be inspected in Figure 11. As is quite typical for RNA-Seq data, here we see that the tagwise dispersion estimates decrease as the tag abundance (log-concentration) increases.

```
> summary(d$tagwise.dispersion)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.007   0.019   0.034   0.056   0.074   0.789

> d$common.dispersion

[1] 0.02

> png(file="Li_tgw-disp_vs_logconc.png", height=600, width=600)
> plot(log2(d$conc$conc.common), d$tagwise.dispersion, panel.first=grid())
> abline(h=d$common.dispersion, col="dodgerblue", lwd=3)
> dev.off()

null device
          1
```

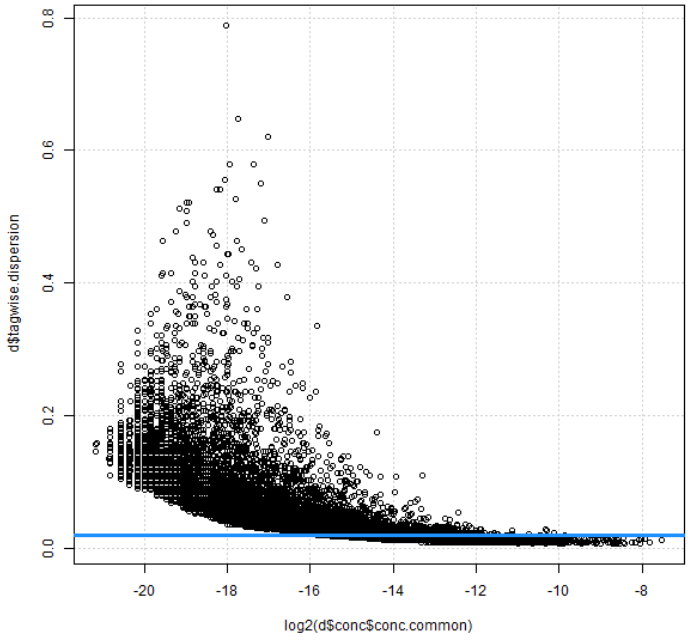

Figure 11: Plot of the tagwise dispersion estimates against abundance (overall expression, here expressed as log-concentration).

12.6.2 Testing

Once we have an estimate of the common dispersion and/or estimates of the tagwise dispersions, we can proceed with testing procedures for determining differential expression using `exactTest`. Here we carry out the testing using the tagwise dispersion estimates calculated using a `prior.n` value of ten.

By default, `exactTest` uses the common dispersion, but by adding the argument `common.disp=FALSE`, tagwise dispersion estimates will be used instead.

```
> de.tgw <- exactTest(d, common.disp = FALSE)

Comparison of groups:  DHT - Control
```

The output below shows that when using tagwise dispersions, the edgeR package still identifies a huge amount of differential expression between the control group and the DHT-treated group. The top DE tags are given even smaller p-values than using the common dispersion—many, many orders of magnitude smaller.

As with the analysis using the common dispersion, all of the top genes have large fold changes, indicating that these changes in expression are likely to be biologically meaningful.

Again, all of the top genes are up-regulated in the DHT-treated group compared with the control group. We note that the ranking of the tags is similar, with seven of the top ten genes using the common dispersion to be found among the top ten genes using tagwise dispersions.

```
> topTags(de.tgw)

Comparison of groups:  DHT-Control
                logConc logFC   P.Value       FDR
ENSG00000096060  -11.32  5.01 2.25e-319 3.71e-315
ENSG00000151503  -11.94  5.82 2.29e-312 1.89e-308
ENSG00000166451  -12.28  4.69 1.23e-213 6.74e-210
ENSG00000127954  -15.62  8.23 4.66e-208 1.92e-204
ENSG00000162772  -10.81  3.32 1.37e-172 4.51e-169
ENSG00000116133  -11.73  3.25 2.32e-148 6.37e-145
ENSG00000113594  -12.83  4.12 6.33e-139 1.49e-135
ENSG00000116285  -13.56  4.21 1.72e-135 3.54e-132
ENSG00000123983  -12.09  3.66 1.20e-129 2.20e-126
ENSG00000115648   -8.82  2.60 1.60e-124 2.64e-121
```

Of course, we can also rank the top tags using the fold change instead of the p-value, as described above.

The tables below shows the quantile-adjusted counts (i.e. counts for equalised library sizes) for the genes that edgeR has identified as the most differentially expressed, using the common dispersion and tagwise dispersions. For these tags, using both methods, there seem to be very large differences between the groups, suggesting that the DE genes identified are truly differentially expressed, and not false positives.

We saw for 't Hoen et al. [2008]'s data how much more consistent the counts *within* groups are for the top tags identified using tagwise dispersions compared with those identified using the common dispersion. This effect is not nearly as pronounced here, as the differences between groups for the top ten tags are so profound (these are after all not true biological replicate samples), but we do note that there is a great deal of consistency in the counts per million within groups for these top tags.

```
> detags.tgw <- rownames(topTags(de.tgw)$table)
> detags.com <- rownames(topTags(de.com)$table)
> cpm.d[detags.tgw, ]

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

```
> cpm.d[detags.com, ]

                lane1  lane2  lane3  lane4 lane5 lane6 lane8
ENSG00000151503  35.8  30.25  33.98  39.71  1814  1875  1795
ENSG00000096060  66.4  68.29  72.81  76.06  2180  2032  2128
ENSG00000127954   0.0   0.00   2.08   2.02   333   328   323
ENSG00000166451  41.9  44.95  39.52  38.37   960   902  1068
ENSG00000131016   9.2   4.32  12.48   4.04   309   206   312
ENSG00000113594  37.8  31.12  39.52  28.94   513   523   613
ENSG00000116285  18.4  24.20  15.95  21.54   354   343   320
ENSG00000123983  63.4  65.70  65.18  72.70   743   686   921
ENSG00000166086   9.2   1.73   2.08   4.04   162   162   177
ENSG00000162772 175.8 176.34 173.35 204.63  1630  1782  1631
```

We might also be interested in comparing the top-ranking genes as identified by edgeR using the common dispersion and tagwise dispersions. We see in the output below that of the top 1000 most DE tags as identified using tagwise dispersions, 86% of these tags are also in the list of the 1000 most DE tags as identified using the common dispersion. This shows that for this dataset there is a great deal of agreement between the common and tagwise dispersion approaches as to which tags are DE.

```
> sum(rownames(topTags(de.tgw, n = 1000)$table) %in% rownames(topTags(de.com,
+      n = 1000)$table))/1000 * 100

[1] 85
```

Using the common dispersion we found that 4936 genes (30% of the total number) are given an adjusted p-value of less than 0.05. In the output below, we see that using tagwise dispersions we obtain slightly fewer DE genes, namely 4438, or 27% of all of the genes in the (filtered) dataset.

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

   [,1]
-1  2179
0  11917
1   2398
```

Of the 4438 tags identified as DE using tagwise dispersions, 2318 (52%) are up-regulated in DHT-treated cells and 2120 (48%) are up-regulated in the control cells. The proportions of up- and down-regulated genes identified using the two approaches to modeling the dispersion are practically equal.

12.6.3 Visualising DGE results

As discussed earlier, the function `plotSmear` can be used to generate a plot of the log-fold change against the log-concentration for each tag. We identify the top 500 most DE tags using both common dispersion and tagwise dispersions so we can highlight them on the plots and compare what we see. The code for producing the fold-change plots (in the one frame for purposes of comparison) is shown below, and the result of this code is shown in Figure 12.

```
> detags500.com <- rownames(topTags(de.com, n = 500)$table)
> detags500.tgw <- rownames(topTags(de.tgw, n = 500)$table)
> png(file="edgeR_case_study_Li_comparative_smearplots.png", height=800, width=600)
> par(mfcol = c(2, 1))
> plotSmear(d, de.tags = detags500.com, main = "Using common dispersion")
> abline(h = c(-2, 2), col = "dodgerblue", lwd = 2)
> plotSmear(d, de.tags = detags500.tgw, main = "Using tagwise dispersions")
> abline(h = c(-2, 2), col = "dodgerblue", lwd = 2)
> dev.off()

null device
          1
```

In Figure 12, the top 500 most differentially expressed genes (those identified as significant by edgeR using the common dispersion (top) and tagwise dispersions (bottom)) are highlighted in red. Looking at Figure 12, we see that, generally speaking, the pattern of differential expression looks similar using the two different methods, and the genes identified as DE have convincingly large fold changes.

We can also look at how well we are modeling the variance in the data by looking at a mean-variance plot. Figure 13 shows the mean-variance plot produced by the plot below.

```
> png(file="edgeR_case_study_Li-meanvarplot.png", height=600, width=600)
> mv <- plotMeanVar(d, show.raw.vars=TRUE, show.tagwise.vars=TRUE,
+                   dispersion.method="qcml", NBline=TRUE)
> dev.off()

null device
          1
```

12.7 Setup

This analysis of Li et al. [2008]'s RNA-seq data was conducted on:

```
> sessionInfo()

R version 2.14.0 Under development (unstable) (2011-06-15 r56138)
Platform: i386-pc-mingw32/i386 (32-bit)
```

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

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

other attached packages:
[1] limma_3.9.5  edgeR_2.3.29

loaded via a namespace (and not attached):
[1] tools_2.14.0
```

and took < 2 minutes to carry out on an Apple MacBook with a 2.8 Ghz Intel Core 2 Duo processor and 8 Gb of 1067 MHz DDR3 memory.

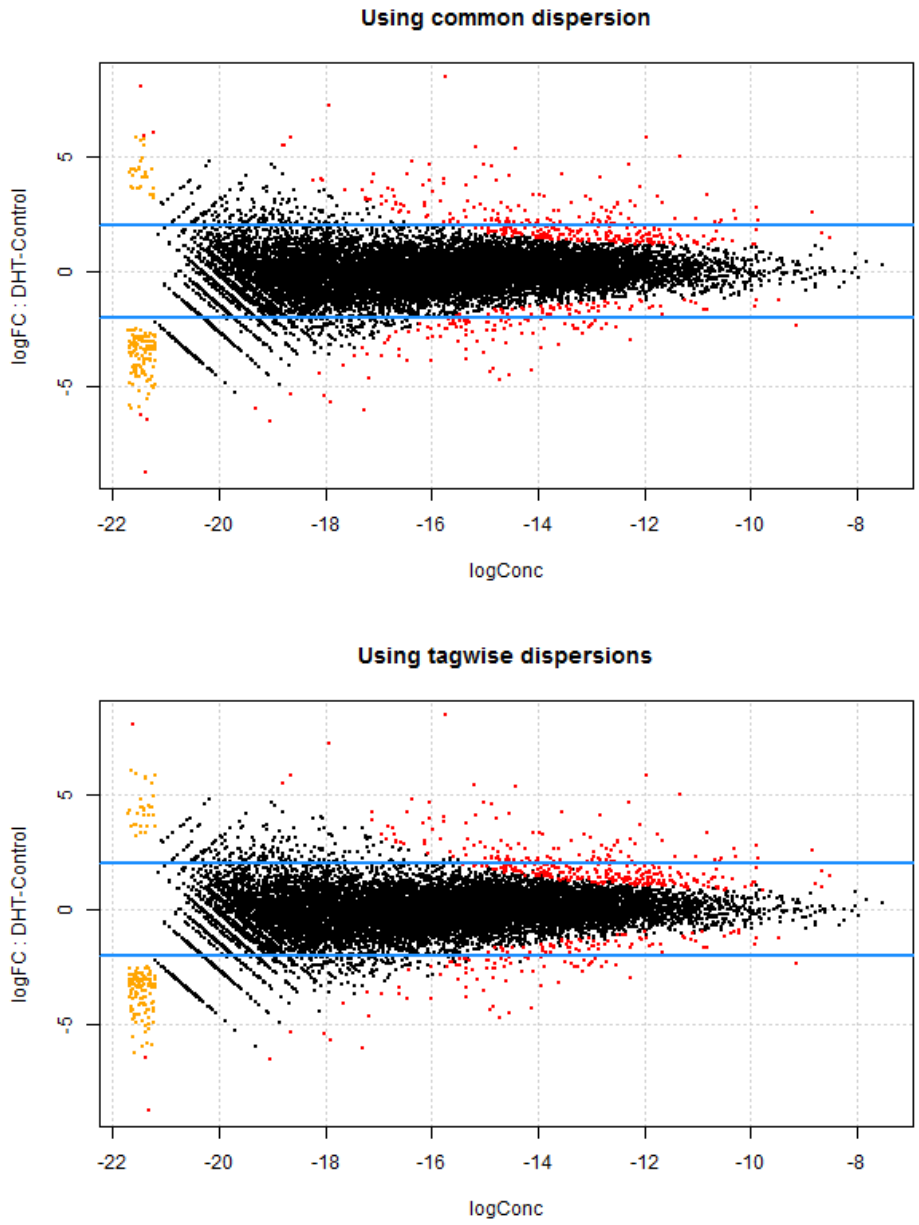

Figure 12: Plots of the log-fold change against the log-concentration for each tag, using the common dispersion (top), and tagwise dispersions (bottom). Tags with positive fold-change here are up-regulated in DHT-treated cells compared with control cells. The 500 most differentially expressed tags according to each method are highlighted in red on both plots.

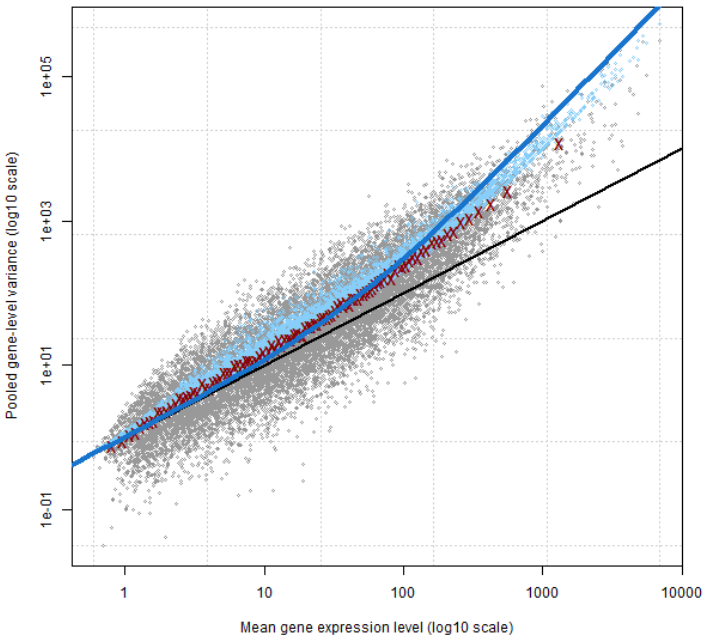

Figure 13: Mean-variance plot showing the raw tagwise variances (grey dots) against tag abundance. The red crosses show the average of raw variance for tags grouped into 100 bins based on overall abundance (averaging is done on the square-root scale to avoid upward bias when these are displayed on the log scale). The light blue dots show the estimated variance for each gene, computed from the tagwise dispersion values. The solid blue line shows the estimated variance using the common dispersion. Overall, the tagwise dispersion estimates look to do a good job of capturing the mean-variance relationship for these data. The black line shows the Poisson variance (variance equals mean). Even for these samples, which are not true biological replicates, the Poisson variance model is inadequate.

13 Case study: Oral carcinomas vs matched normal tissue

13.1 Introduction

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

13.2 Source of the data

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

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

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

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

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

alias2SymbolTable from the limma package) and that we use only the transcript with the greatest number of exons to represent each gene.

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

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

```
> library(edgeR)
> library(limma)
> rawdata <- read.csv(file="tuch_counts.csv", stringsAsFactors=FALSE)
> head(rawdata)

          X        X.1           X.2    X8N    X8T   X33N   X33T   X51N   X51T
1                                    counts counts counts counts counts counts
2  idRefSeq nameOfGene numberOfExons    sum    sum    sum    sum    sum    sum
3 NM_182502  TMPRSS11B            10   2592      3   7805    321   3372      9
4 NM_003280      TNNC1             6   1684      0   1787      7   4894    559
5 NM_152381      XIRP2            10   9915     15  10396     48  23309   7181
6 NM_022438        MAL             3   2496      2   3585    239   1596      7

> library(org.Hs.eg.db)
> rawtable <- rawdata[-c(1,2),]
> refseqid <- as.character(rawtable[,1])
> head(refseqid)

[1] "NM_182502"    "NM_003280"    "NM_152381"    "NM_022438"    "NM_001100112"
[6] "NM_017534"

> idfound <- refseqid %in% mappedRkeys(org.Hs.egREFSEQ)
> table(idfound)

idfound
FALSE  TRUE
  313 15355

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

[1] "TMPRSS11B" "TNNC1"     "XIRP2"     "MAL"       "MYH2"      "MYH2"

> length(genes)
```

```
[1] 15317

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

[1] 10  6 10  3 40 40

> length(nexons)

[1] 15317

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

               N8 T8   N33 T33   N51  T51
NM_182502    2592  3  7805 321  3372    9
NM_003280    1684  0  1787   7  4894  559
NM_152381    9915 15 10396  48 23309 7181
NM_022438    2496  2  3585 239  1596    7
NM_001100112 4389  7  7944  16  9262 1818
NM_017534    4402  7  7943  16  9244 1815

> dim(counts)

[1] 15317     6

> o <- order(nexons, decreasing=TRUE)
> counts.ord <- counts[o,]
> genes.ord <- genes[o]
> keep <- !duplicated(genes.ord)
> sum(keep)

[1] 10464

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

An object of class "DGEList"
$samples
     group lib.size norm.factors
N8  normal  7795290        1.157
T8  tumour  7205310        1.091
```

```
N33 normal 15761188        0.662
T33 tumour 14070267        0.958
N51 normal 21083214        1.039
T51 tumour 14819300        1.204

$counts
             N8  T8   N33  T33   N51  T51
NM_000014  2242 261  2285  597 15121 1991
NM_144670 11731 912 13308 3071  6944 1160
NM_017436   162 296   111  362   751  182
NM_015665   199  81   215  344   512  342
NM_023928   470 710   573 1112   690  728
10459 more rows ...

$genes
[1] "A2M"    "A2ML1"  "A4GALT" "AAAS"   "AACS"
10459 more rows ...

$all.zeros
NM_000014 NM_144670 NM_017436 NM_015665 NM_023928
    FALSE     FALSE     FALSE     FALSE     FALSE
10459 more elements ...
```

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

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

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

[1] 10464
```

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

13.4 Producing an MDS plot

Before proceeding with the computations for differential expression, it is possible to produce a plot showing the sample relations based on multidimensional scaling. The function `plotMDS.dge` produces an MDS plot for the samples when provided with the `DGEList` object, as shown in Figure 14.

```
> pdf(file="edgeR_case_study_Tuch_MDSplot.pdf", height=6, width=6)
> plotMDS(d.tuch, main="MDS Plot for Tuch Data")

Using grid search to estimate tagwise dispersion.

> dev.off()

null device
          1

> tools::compactPDF("edgeR_case_study_Tuch_MDSplot.pdf")

NULL
```

From the MDS plot, it can be seen that the libraries T33 and T8 (tumour samples from patients 33 and 8 respectively) are most different from the other samples, but we will not remove them from the analysis as we will just be demonstrating the use of edgeR.

13.5 The design matrix

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

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

    (Intercept) patient33 patient51 tumour
N8            1         0         0      0
T8            1         0         0      1
N33           1         1         0      0
T33           1         1         0      1
N51           1         0         1      0
T51           1         0         1      1
attr(,"assign")
```

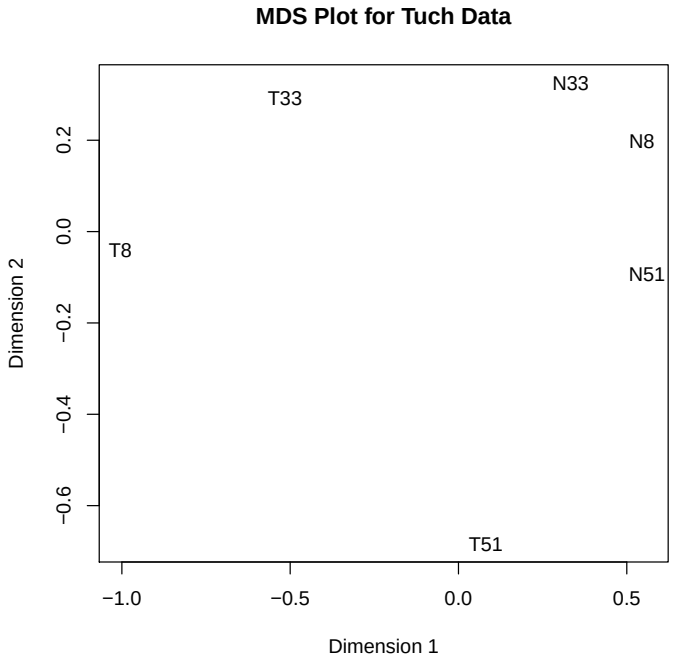

Figure 14: Multidimensional scaling (MDS) plot for the Tuch data, showing the relations between the samples in two dimensions. From this plot, the samples T33 and T8 can be identified easily as outliers—there is a large distance between these two samples and the others.

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

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

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

13.6 Analysis using Cox-Reid common dispersion

13.6.1 Estimating the Cox-Reid common dispersion

The first major step in the analysis of DGE data using the NB model is to estimate the dispersion parameter for each tag. Note that this is a paired design experiment, so the

dispersion has to be estimated in a different way such that both the cell-type and the patient factors are taken into account.

Like the qCML method (i.e.,the `estimateCommonDisp()` and the `estimateTagwiseDisp()` function) we used in previous case studies, the CR method also calculates both the common dispersion and tagwise dispersions. The most straight-forward analysis for a paired design experiment uses the CR common dispersion estimate as the dispersion for all tags. For many applications this will be adequate and it may not be necessary to estimate the CR tagwise dispersions, i.e. estimate the CR dispersion separately for each tag.

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

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

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

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

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

```
> d.tuch$common.dispersion

[1] 0.16

> sqrt(d.tuch$common.dispersion)

[1] 0.4
```

13.6.2 Testing

Once we have an estimate of the CR common dispersion, we can proceed with testing procedures for determining differential expression. Since this is a paired design experiment, we have to use the new testing method, the GLM method, rather than the exact test (the one we demonstrated in the previous case studies).

The GLM method fits a negative binomial generalized linear model for each gene/tag with the unadjusted counts provided, a value for the dispersion parameter and, optionally, offsets and weights for different libraries or transcripts. This is done using the funtion `glmFit()` and `glmLRT()`.

The function `glmFit()` calls the in-built function `glm.fit()` to fit the NB GLM for each tag and produces an object of class `DGEGLM`. Once we have a fit for a given design matrix, `glmLRT()` can be run with a given coefficient or contrast specified and evidence for differential expression can be assessed using a likelihood ratio test. The `glmLRT` function produces an object of class `DGELRT` with a table containing the abundance of each tag (log-concentration, logConc), the log-fold change of expression between conditions/contrasts being tested (logFC), the likelihood ratio statistic (LR.statistic) and the p-value from the LR test (p.value), for each tag in the dataset. Then tags can be ranked in order of evidence for differential expression, based on either the p-value or the log-fold change of expression computed for each tag.

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

```
> glmfit.tuch <- glmFit(d.tuch, design,
+                       dispersion = d.tuch$common.dispersion)
> lrt.tuch <- glmLRT(d.tuch, glmfit.tuch, coef=4)
> topTags(lrt.tuch)

Coefficient:  tumour
                 genes logConc logFC    LR  P.Value      FDR
NM_182502    TMPRSS11B   -8.51 -7.32 121.9 2.46e-28 2.57e-24
NM_016190         CRNN   -6.99 -7.26 107.8 3.05e-25 1.59e-21
NM_002371          MAL   -8.95 -6.81 106.7 5.29e-25 1.84e-21
NM_002465       MYBPC1   -8.16 -7.02 100.0 1.51e-23 3.94e-20
NM_014440        IL1F6  -10.06 -6.15  96.3 9.84e-23 2.06e-19
NM_002272         KRT4   -5.62 -7.13  93.1 5.09e-22 8.24e-19
NM_001010909     MUC21   -8.80 -6.72  92.9 5.51e-22 8.24e-19
NM_001100        ACTA1   -8.39 -6.28  92.3 7.34e-22 9.60e-19
NM_003280        TNNC1   -9.19 -6.96  91.6 1.05e-21 1.23e-18
NM_006063      KBTBD10   -8.23 -6.21  89.3 3.37e-21 3.53e-18
```

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

The table below shows the raw counts for the tags that edgeR has identified as the most differentially expressed. For these tags there seems to be very large differences between the

groups, suggesting that the DE tags identified are truly differentially expressed, and not false positives.

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

               N8   N33   N51  T8   T33  T51
NM_182502    2592  7805  3372   3   321    9
NM_016190   24146 22026 12480  49  2353   26
NM_002371    2697  3941  1750   3   265    8
NM_002465    4809  4146 15623  10    14 1311
NM_014440     367  1824   802  10    45    1
NM_002272   76461 99082 47411 353 20651   31
NM_001010909 4160  3425  1720   7   516    5
NM_001100    3334  3198 13643   8    32 1063
NM_003280    1684  1787  4894   0     7  559
NM_006063    4325  3115 16007  24    17 1461
```

Note that the 2nd tag ('CKM') and the 7th tag ('MYBPC1') have much larger counts in patient 55 than in the other two patients, which shows that the effect from the patients does exist and the GLM method can pick that up.

If we order the genes by fold change instead of p-value, as in the table below, we see that the tags with the largest fold changes have very small concentrations. This ranking is dominated by genes that have zero counts in one group and is less informative than ranking by p-value.

```
> topTags(lrt.tuch, sort.by = "logFC")

Coefficient:  tumour
                genes logConc logFC    LR  P.Value      FDR
NM_001100112     MYH2   -8.13 -7.35  87.2 9.84e-21 9.36e-18
NM_182502    TMPRSS11B   -8.51 -7.32 121.9 2.46e-28 2.57e-24
NM_016190         CRNN   -6.99 -7.26 107.8 3.05e-25 1.59e-21
NM_002272         KRT4   -5.62 -7.13  93.1 5.09e-22 8.24e-19
NM_002465       MYBPC1   -8.16 -7.02 100.0 1.51e-23 3.94e-20
NM_003280        TNNC1   -9.19 -6.96  91.6 1.05e-21 1.23e-18
NM_152381        XIRP2   -7.43 -6.93  76.1 2.74e-18 8.98e-16
NM_002371          MAL   -8.95 -6.81 106.7 5.29e-25 1.84e-21
NM_001010909     MUC21   -8.80 -6.72  92.9 5.51e-22 8.24e-19
NM_198060         NRAP   -8.45 -6.44  82.9 8.54e-20 4.96e-17

> top <- rownames(topTags(lrt.tuch, sort.by = "logFC")$table)
> d.tuch$counts[top,order(d.tuch$samples$group)]

               N8   N33   N51  T8   T33  T51
NM_001100112 4389  7944  9262   7    16 1818
NM_182502    2592  7805  3372   3   321    9
NM_016190   24146 22026 12480  49  2353   26
```

```
NM_002272    76461 99082 47411 353 20651   31
NM_002465     4809  4146 15623  10    14 1311
NM_003280     1684  1787  4894   0     7  559
NM_152381     9915 10396 23309  15    48 7181
NM_002371     2697  3941  1750   3   265    8
NM_001010909  4160  3425  1720   7   516    5
NM_198060     3741  1990 12531   4    17 1829
```

We see in the output below that over 1200 tags are significantly differentially expressed according to edgeR when using the CR common dispersion estimate and GLM likelihood ratio test. Of those tags, 298 are up-regulated in the tumour tissues compared with the normal tissues and 976 are down-regulated in the tumour tissues compared with normal tissues.

```
> summary(decideTestsDGE(lrt.tuch))

   [,1]
-1  976
0  9190
1   298
```

13.7 Cox-Reid dispersions with mean-dependent trend

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

```
> d.tuch <- estimateGLMTrendedDisp(d.tuch, design)
> summary(d.tuch$trended.dispersion)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.121   0.130   0.151   0.160   0.171   0.388
```

An analysis could be carried out just as for the common dispersion above, but is not shown here.

13.8 Analysis using Cox-Reid tagwise dispersion

13.8.1 Estimating the Cox-Reid tagwise dispersion

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

As noted earlier, the dispersion parameter is the overdispersion relative to the Poisson, and represents the biological, or sample-to-sample variability. The methods we have developed moderate the dispersion estimates towards a common dispersion, much like how the limma package moderates the variances in the analysis of microarray data.

The amount of moderation done is determined by the value of a weight parameter `prior.n`. The value for `prior.n` corresponds to the number of individual tags equivalent to the weight given to the common likelihood. Thus, the higher `prior.n`, the more strongly the individual dispersion estimates are moderated, or 'squeezed', towards the common value. To run the moderated analysis, we need to determine how much moderation is necessary. How best to do this is still an open research question, but we currently recommend selecting a value for the weight parameter `prior.n` *a priori* and have found that very good results can be obtained this way.

In our experience analysing RNA-Seq data we have found that a good rule of thumb for choosing a value for `prior.n` is to choose a certain number of prior degrees of freedom (a value between 20 and 30 works well) and then divide this number by the degrees of freedom (number of samples minus the number of variables being fit in the model; in the case of a simple multiple-group comparison this is simply the number of samples minus the number of groups). The function `getPriorN` automatically calculates the appropriate value for `prior.n` for a given experimental design.

In an experiment such as that we consider here, in which we have just six samples, with two groups (group factor) and three patients (blocking factor), and we have just two degrees of freedom for estimating the dispersion parameter. Standard tagwise dispersion estimates are likely to be unreliable, so we want to give a reasonable weight to the common likelihood. We need to choose a value for `prior.n` such that individual tagwise dispersion estimates are 'squeezed' quite strongly towards the common dispersion. As noted, we have two degrees of freedom from the experimental design. Thus, setting the `prior.n` to be 10 (20 divided by two) should be appropriate. This means that the common likelihood receives the weight of ten individual tags. Therefore, there will be a reasonable degree of 'squeezing' towards the common dispersion estimate, but still enough scope to allow flexibility when estimating the individual dispersion for each gene. By default, `estimateTagwiseDisp` uses the `prior.n` value from `getPriorN`, which in this case is `prior.n=10`..

The function `estimateTagwiseDisp` produces a DGEList object that contains all of the elements present in the object produced by `estimateCommonDisp`, as well as the value for `prior.n` used (`d$prior.n`) and the tagwise dispersion estimates (`d$tagwise.dispersion`), as we see below. We use the `trend` argument below to indicate that we wish to allow a dependence of the dispersion on the overall expression level.

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

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

```
 [7] "abundance"          "bin.dispersion"     "bin.abundance"
[10] "tagwise.dispersion"
```

```
> head(d.tuch$tagwise.dispersion)
```

```
NM_000014 NM_144670 NM_017436 NM_015665 NM_023928 NM_024666
    0.176     0.260     0.192     0.154     0.122     0.121
```

It is interesting to consider the distribution of the CR tagwise dispersion estimates. As we can see from the output below, the CR tagwise dispersion estimates range from a minimum of 0.11 to a maximum of 0.46. The range of dispersions is therefore large, but the tags in the middle two quartiles of the CR tagwise dispersion estimates have dispersion estimates close to the CR common dispersion estimate.

```
> summary(d.tuch$tagwise.dispersion)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.109   0.127   0.148   0.162   0.173   0.585
```

```
> png("edgeR_case_study_Tuch_BCV_vs_abundance.png", height=600, width=600)
> plot(d.tuch$abund+log(1e06), sqrt(d.tuch$tagwise.dispersion), xlab="Counts per million (log scale)", yl
> oo <- order(d.tuch$abundance)
> lines(d.tuch$abundance[oo]+log(1e06), sqrt(d.tuch$trended.dispersion[oo]), col="dodgerblue", lwd=3)
> abline(h=sqrt(d.tuch$common.dispersion), col="firebrick", lty=3, lwd=3)
```

13.8.2 Testing

The testing procedures when using CR tagwise dispersion estimates are carried out exactly as for the CR common dispersion, as described above. Here we carry out the testing using the CR tagwise dispersion estimates calculated using a `prior.n` value of eight. The GLM fit and the likelihood ratio test are done using the same functions as before (i.e. `glmFit()` and `glmLRT()`), the only difference is that we use CR tagwise dispersions as the dispersion in the `glmFit()` function.

```
> glmfit.tuch.tgw <- glmFit(d.tuch, design,
+                           dispersion = d.tuch$tagwise.dispersion)
> lrt.tuch.tgw <- glmLRT(d.tuch, glmfit.tuch.tgw)
```

The output below shows that when using CR tagwise dispersions, the edgeR package still identifies a lot of differential expression between the normal tissue group and the tumour tissue group. This arises because the moderated tagwise dispersions can be much smaller than the common dispersion, and tags with smaller dispersions will have smaller p-values than the same tags with p-values computed using a common dispersion. As with the analysis using the common dispersion, all of the top tags have a large fold change, indicating that these changes in expression are likely to be biologically meaningful. We note that the ranking is different, however, and not all of the top ten tags according to using the common dispersion are found to be among the top ten tags using tagwise dispersions.

```
> options(digits = 4)
> topTags(lrt.tuch.tgw)

Coefficient:  tumour
                 genes logConc  logFC     LR   P.Value       FDR
NM_014440        IL1F6 -10.060 -6.129 110.00 9.812e-26 1.027e-21
NM_182502    TMPRSS11B  -8.506 -7.386  96.62 8.414e-23 4.402e-19
NM_005609         PYGM  -9.653 -5.474  92.42 7.001e-22 2.442e-18
NM_001039585     PTGFR -10.520 -5.195  90.21 2.144e-21 5.608e-18
NM_057088         KRT3  -9.301 -5.818  83.15 7.624e-20 1.595e-16
NM_004533       MYBPC2  -9.299 -5.461  81.06 2.186e-19 3.813e-16
NM_002371          MAL  -8.952 -6.890  80.26 3.278e-19 4.900e-16
NM_004320       ATP2A1  -9.669 -4.621  79.95 3.831e-19 5.011e-16
NM_001111283      IGF1  -9.841 -3.992  76.98 1.725e-18 2.005e-15
NM_001976         ENO3  -9.423 -5.175  75.17 4.325e-18 4.526e-15
```

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

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

               N8  N33  N51 T8 T33 T51
NM_014440     367 1824  802 10  45   1
NM_182502    2592 7805 3372  3 321   9
NM_005609    1399 1267 2171 22  16 103
NM_001039585  455  287 1736  7  12  46
NM_057088    1069 3774  885  7 358   5
NM_004533     966  486 8045 11   6 457
NM_002371    2697 3941 1750  3 265   8
NM_004320     988 1558 2285 25  52 161
NM_001111283  460  343 4703 25  26 257
NM_001976    1092 1292 4841  4  74 223
```

We see in the output below that 1269 tags are significantly differentially expressed according to edgeR when using the CR tagwise dispersion estimate and GLM likelihood ratio test. It is slightly less the total number of DE tags under the CR common dispersion method. Of those 1269 tags, 307 are up-regulated in the tumour tissues compared with the normal tissues and 962 are down-regulated in the tumour tissues compared with normal tissues.

```
> summary(decideTestsDGE(lrt.tuch.tgw))

   [,1]
-1  957
0  9196
1   311
```

13.9 Setup

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

```
> sessionInfo()

R version 2.14.0 Under development (unstable) (2011-06-15 r56138)
Platform: i386-pc-mingw32/i386 (32-bit)

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

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

other attached packages:
[1] org.Hs.eg.db_2.5.0   RSQLite_0.9-4        DBI_0.2-5
[4] AnnotationDbi_1.15.9 Biobase_2.13.6       limma_3.9.5
[7] edgeR_2.3.29

loaded via a namespace (and not attached):
[1] tools_2.14.0
```

References

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

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

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

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

DJ McCarthy, Yunshun Chen, and GK Smyth. Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. *Submitted*, 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. URL `http://genomebiology.com/2010/11/3/R25`.

Mark D Robinson, Davis J McCarthy, and Gordon K Smyth. edgeR: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1): 139–40, Jan 2010. doi: 10.1093/bioinformatics/btp616. URL `http://bioinformatics.oxfordjournals.org/cgi/content/full/26/1/139`.

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

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

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