edgeR: differential expression analysis of digital gene expression data

Mark Robinson
mrobinson@wehi.edu.au

Davis McCarthy
dmccarthy@wehi.edu.au

Yunshun Chen
yuchen@wehi.edu.au

Gordon K. Smyth

14 Oct 2010

1 Introduction

This document gives an introduction and overview of the R Bioconductor package edgeR [Robinson et al., 2010], which provides statistical routines for determining differential expression in digital gene expression data. The package implements methods developed by Robinson and Smyth [2007, 2008]. The routines can be applied equally to RNA-seq, Tag-seq, SAGE, CAGE, Illumina/Solexa, 454 or ABI SOLiD experiments. In fact, the methods may be useful in other experiments where counts are observed.

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`. 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 helpful posting guide at `http://www.bioconductor.org/doc/postingGuide.html`.

3 Reading data

edgeR requires three pieces of information:

1. `counts`: a matrix of counts where each row represents a gene/exon (or whatever genomic feature is being tracked) and each column is a different sample. The row names are transcript IDs.

2. `group`: a factor (with length equal to the number of columns of `counts`) denoting the experimental group.

3. `lib.size`: vector of the same length as `group` giving the total number of reads sequenced for each sample.

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 appriopriate in-built `R` function (such as `read.delim` or `read.csv`) can be used to read the table of counts into `R`. The library sizes can be the column sums from the table of counts, and thus easily obtained, or the user can specify the the library sizes throught `lib.size` argument of the `DGEList()` constructor. 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.

4 Normalization issues for digital counts

4.1 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.

4.2 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")
```

5 Negative binomial models

The basic model we use for DGE data is based on the negative binomial distribution. The model is very flexible. For example, if Y is distributed as $NB(\mu, \phi)$, then the expected value of Y is μ and the variance is $\mu + \mu^2 \cdot \phi$, thus giving sufficient flexibility for many scenarios in observing count data.

The observed data can be denoted as Y_{gij} where g is the gene (tag, exon, etc.), i is the experimental group and j is the index of the replicate. We can model the counts as

$$Y_{gij} \sim NB(M_j \cdot p_{gi}, \phi_g)$$

where p_{gi} represents the proportion of the sequenced sample for group i that is tag g and M_j represents the library size.

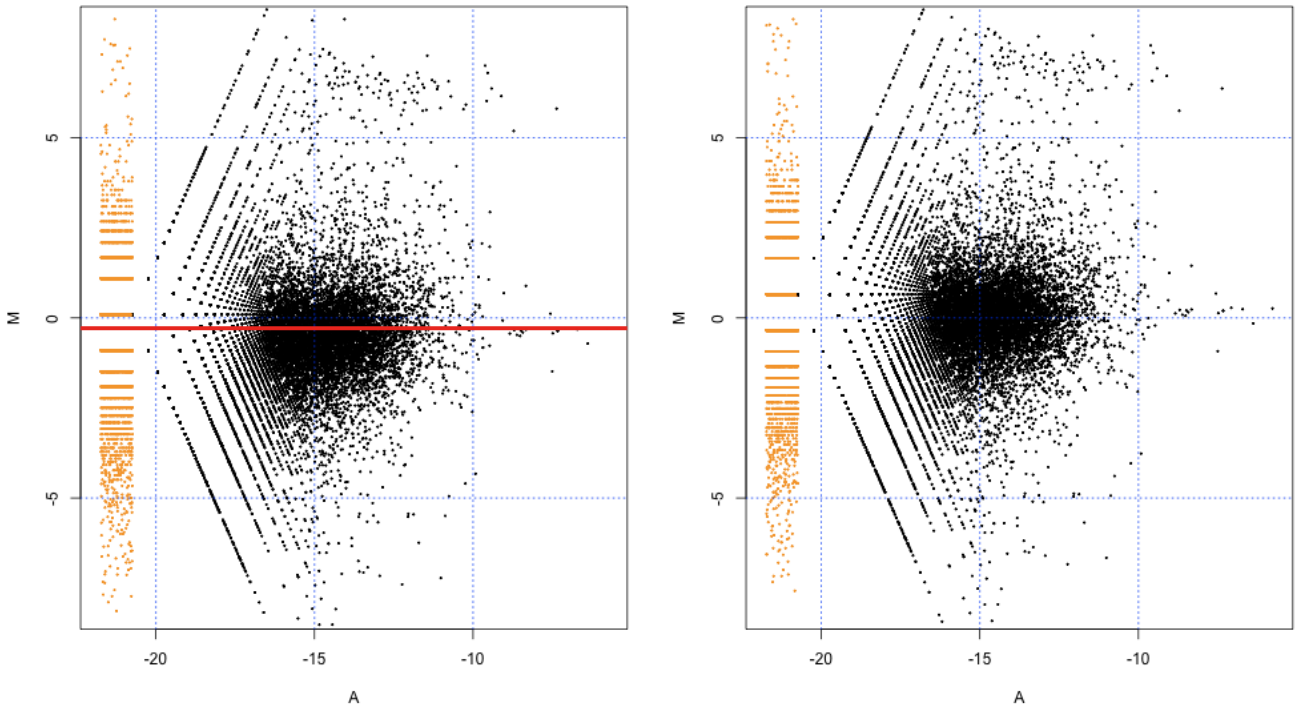

Figure 1: Smear plots before (left) and after (right) composition normalization.

It is of interest to find genes where, for example, p_{g1} is significantly different from p_{g2}. The parameter ϕ_g is the overdispersion (relative to the Poisson) and represents the biological, or sample-to-sample variability. The methods we developed moderate the dispersion estimates towards a common dispersion, much like how the limma package moderates the variances in the analysis of microarray data. It is also possible to analyse DGE data using a common dispersion for each tag using edgeR.

6 Estimating dispersions

6.1 Two ways of estimating dispersions

When a negative binomial model is fitted, we need to estimate the dispersion(s) before we carry out the analysis. edgeR provides two ways of estimating the dispersion(s), the quantile-adjusted conditional maximum likelihood (qCML) method and the Cox-Reid profile-adjusted likelihood (CR) method. In general, we apply the qCML method to experiments with single factor and the CR method to experiments with multiple factors.

6.2 Experiment 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.

For a single tag with a small number of libraries, all estimators offer mediocre performance and here is no clear winner. As the number of tags used to estimate the common dispersion increases while holding the number of libraries at a small number, qCML is clearly the best estimator. With more libraries, CR method performs about as well as qCML.

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.

For more detailed examples, see the case studies in section 8 (Zhang's data), section 9 ('t Hoen's data) and section 10 (Li's data)

6.3 Experiment with multiple factors

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 common dispersion and tagwise dispersions can be calculated using the `estimateCRDisp()` function (for tagwise dispersions, set `'tagwise = TRUE'` within the function), and it is strongly recommended in multi-factor experiment cases.

For more detailed examples, see the case study in section 11 (Tuch's data).

6.4 Tagwise dispersion or common dispersion

edgeR can estimate a common dispersion for all the tags or it can estimate separate dispersions for individual tags. As individual tags typically don't provide enough data to estimate the dispersion reliably, edgeR implements 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 50, 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=25`. This is an empirical rule of thumb borne out of experience with a number of datasets.

7 Testing for DE genes/tags

7.1 Two ways of testing for differential expression

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. edgeR provides two ways of testing differential expression, the exact test and the generalized linear model (GLM) likelihood ratio test.

7.2 Experiment with single factor

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 more detailed examples, see the case studies in section 8 (Zhang's data), section 9 ('t Hoen's data) and section 10 (Li's data).

7.3 Experiment with multiple factors

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.

For more detailed examples, see the case study in section 11 (Tuch's data).

8 Case study: SAGE data

8.1 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].

8.2 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.

8.3 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)
> setwd("/Users/davismcc/Documents/Honours/Data/ZhangData")
> 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
GSM728 GSM728.txt    NC               Normal colon    50179
GSM729 GSM729.txt    NC               Normal colon    49593
GSM755 GSM755.txt    Tu Primary colonrectal tumour    57686
GSM756 GSM756.txt    Tu Primary colonrectal tumour    49064

$counts
           GSM728 GSM729 GSM755 GSM756
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 ...
```

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.

8.4 Analysis using common dispersion

8.4.1 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"             "conc"
[7] "common.lib.size"
```

The output of estimateCommonDisp is a DGEList object with several new elements. The element common.dispersion, as the name suggests, provides the estimate of the common dispersion, 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.

We see in the output below that the total number of counts in each library of the pseudocounts agrees well with the common library size, as desired.

```
> d$samples$lib.size

[1] 50179 49593 57686 49064

> d$common.lib.size

[1] 51516

> colSums(d$pseudo.alt)

GSM728 GSM729 GSM755 GSM756
 51512  51513  51674  51483
```

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 0.45. 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, 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] 0.1988

> sqrt(d$common.dispersion)

[1] 0.4458
```

8.4.2 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.

```
> options(digits = 4)
> topTags(de.com)
```

```
Comparison of groups:  Tu - NC
            logConc   logFC    PValue       FDR
AGCTGTTCCC  -28.24  43.552 1.017e-19 5.841e-15
CTTGGGTTTT  -29.88  40.271 2.967e-10 8.522e-06
TACAAAATCG  -30.26  39.510 2.712e-08 4.926e-04
CCCAACGCGC  -12.80  -5.849 3.430e-08 4.926e-04
GCCACCCCCT  -30.33  39.380 6.341e-08 7.285e-04
CCAGTCCGCC  -30.43  39.177 1.844e-07 1.513e-03
GTCATCACCA  -30.42 -39.189 1.844e-07 1.513e-03
TCACCGGTCA  -11.13  -4.229 4.428e-07 3.180e-03
TAAATTGCAA  -11.39  -4.242 6.090e-07 3.887e-03
CGCGTCACTA  -12.30   4.611 7.300e-07 4.194e-03
```

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, ]
```

```
           GSM728 GSM729 GSM755 GSM756
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
TCACCGGTCA    118     75      6      5
TAAATTGCAA    103     59      3      6
CGCGTCACTA      1      3     88     21
```

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    PValue       FDR
AGCTGTTCCC  -28.24  43.55 1.017e-19 5.841e-15
CTTGGGTTTT  -29.88  40.27 2.967e-10 8.522e-06
TACAAAATCG  -30.26  39.51 2.712e-08 4.926e-04
GCCACCCCCT  -30.33  39.38 6.341e-08 7.285e-04
GTCATCACCA  -30.42 -39.19 1.844e-07 1.513e-03
CCAGTCCGCC  -30.43  39.18 1.844e-07 1.513e-03
GTGCGCTGAG  -30.67  38.69 2.362e-06 9.048e-03
CTTGACATAC  -30.69 -38.66 2.837e-06 9.588e-03
GTGTGTTTGT  -30.72  38.59 4.135e-06 1.320e-02
GGGGGGGGGG  -30.74  38.56 5.019e-06 1.518e-02
```

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 264 genes are significantly differentially expressed according to edgeR when using the common dispersion estimate. Of those genes, 100 are up-regulated in the cancer cells compared with the normal cells and 164 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] 264
```

```
> top264 <- topTags(de.com, n = 264)
> sum(top264$table$logFC > 0)
```

```
[1] 100
```

```
> sum(top264$table$logFC < 0)
```

```
[1] 164
```

Furthermore, we see below that 33 tags (0.06% 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.

```
> sum(p.adjust(de.com$table$p.value, method = "BH") < 0.05)
```

```
[1] 33
```

```
> mean(p.adjust(de.com$table$p.value, method = "BH") < 0.05) *
+     100
```

```
[1] 0.05744
```

8.4.3 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.

```
> detags264 <- rownames(topTags(de.com, n = 264)$table)
```

```
> plotSmear(d, de.tags = detags264, main = "FC plot using common dispersion")
> abline(h = c(-2, 2), col = "dodgerblue")
```

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.

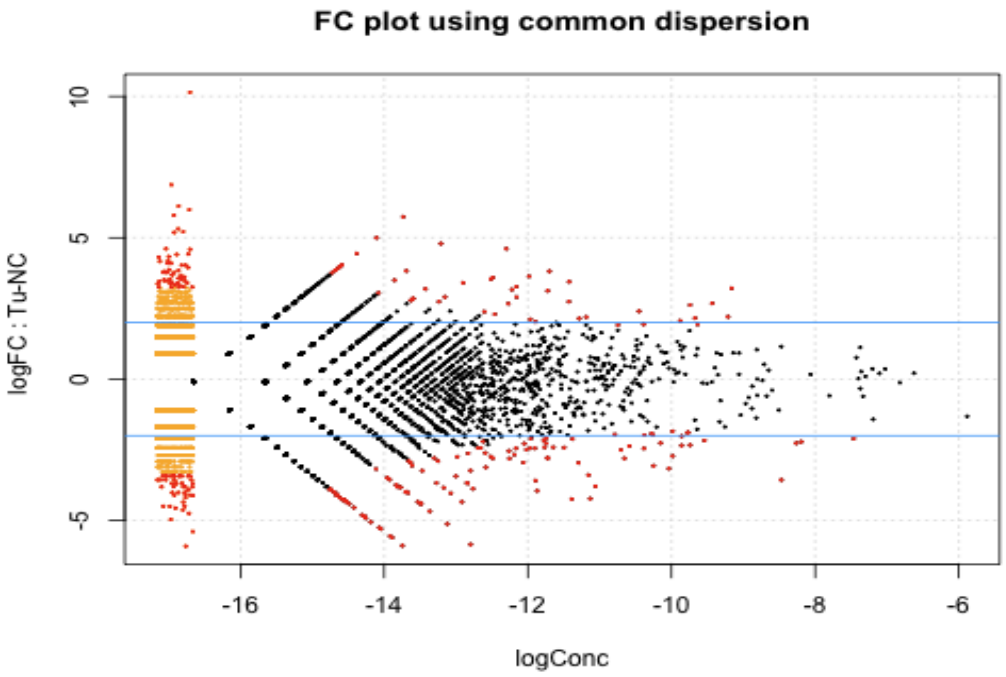

Figure 2: Plot of the log-fold change against the log-concentration for each tag. The 264 most differentially expressed tags as identified by edgeR using the common dispersion are outlined in red.

8.5 Analysing the data using moderated tagwise dispersions

8.5.1 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 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 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 an experiment such as that we consider here, in which we have just four samples, two in each group, and thus 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. Here, we choose a moderate amount of smoothing—we let `prior.n` be 10. This means that the common likelihood receives the weight of 10 individual tags, so there will be a reasonable degree of 'squeezing', but there is still ample scope to estimate an individual dispersion for each gene.

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.

```
> d <- estimateTagwiseDisp(d, prior.n = 10)

Using grid search to estimate tagwise dispersion.

> names(d)

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

> d$prior.n

[1] 100

> head(d$tagwise.dispersion)

[1] 2.612336 0.065168 0.001001 0.268445 0.053908 0.112720
```

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 0.001 to a maximum of 3.41. The range of dispersions is therefore large, but the tags in the middle two quartiles of the tagwise dispersion estimates have dispersion estimates just slightly smaller than the common dispersion estimate.

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

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.0010 0.1926 0.1998 0.2132 0.1998 3.4120

> d$common.dispersion

[1] 0.1988
```

8.5.2 Estimating smoothing using an approximate eBayes rule

For this, we can use an empirical Bayes rule that involves calculating a weight parameter prior.n. However, for many applications (especially if the estimated smoothing parameter is large), using the common dispersion for all tags will give excellent results.

In order to determine how much moderation is necessary, we require an estimate of the dispersion parameter, so we use the common dispersion estimate from `estimateCommonDisp`. The smoothing parameter, `prior.n` can then be calculated. As we see below, for this dataset the estimate of the smoothing parameter is very large. This estimate arises from the fact that the estimate of the overdispersion of the likelihood scores (a critical aspect of the algorithm for estimating the amount of smoothing required) is strictly zero. In this circumstance the algorithm is indicating that we should be using a common dispersion, and so returns a very large suggested value for `prior.n`. If we were to use this large value for the weight parameter we would moderate the tagwise dispersion estimates so strongly that it would be equivalent to using the common dispersion estimate.

```
> prior.n <- estimateSmoothing(d)

  Warning message:
In .odls(scores.g = scores, info.g = exp.inf) :
  Estimate of overdispersion of likelihood scores is strictly zero.
 Returning min val close to zero (1e-08).

> prior.n

[1] 182665666
```

8.5.3 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—indeed the p-values of the top tags are even smaller than the top tags based on the common dispersion. 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.

```
> topTags(de.tgw)
```

```
Comparison of groups:  Tu - NC
           logConc  logFC    PValue       FDR
CTAAGACTTC  -7.191 -1.419 3.610e-56 2.074e-51
TGCTCCTACC  -9.985 -2.722 2.690e-21 7.726e-17
GTGCTGAATG -10.385 -2.459 6.358e-18 1.218e-13
CGCTGTGGGG -11.303 -2.982 2.311e-16 3.319e-12
GGGGTCAGGG  -9.639  1.934 1.928e-15 2.215e-11
GATGACCCCC -11.900 -3.603 4.097e-15 3.923e-11
TTATGGGATC -11.724  3.122 8.147e-14 6.065e-10
TCACCGGTCA -11.130 -4.229 8.446e-14 6.065e-10
ATGCGGGAGA -11.720 -2.797 5.618e-12 3.586e-08
GGCTGGGGGC -10.209 -1.701 1.012e-11 5.816e-08
```

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, ]
```

```
  GSM728 GSM729 GSM755 GSM756
CTAAGACTTC    559    558    248    199
TGCTCCTACC    140    113     22     19
GTGCTGAATG     96     79     17     17
CGCTGTGGGG     58     53      6      9
GGGGTCAGGG     27     37    134    127
GATGACCCCC     42     49      3      5
TTATGGGATC      6      4     48     45
TCACCGGTCA    118     75      6      5
ATGCGGGAGA     39     39      6      6
GGCTGGGGGC     75     77     26     24
```

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

We see in the output below that 309 genes are significantly differentially expressed according to edgeR when using the tagwise dispersion estimates (ten fewer than when using the common dispersion). Of those tags, 100 are up-regulated in the cancer cells compared with the normal cells and 209 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] 309

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

[1] 100

> sum(toptgw$table$logFC < 0)

[1] 209
```

Furthermore, we see below that 72 tags (0.13% 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.

```
> sum(p.adjust(de.tgw$table$p.value, method = "BH") < 0.05)

[1] 72

> mean(p.adjust(de.tgw$table$p.value, method = "BH") < 0.05) *
+     100

[1] 0.1253307
```

8.5.4 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 3.

```
> detags.tgw <- rownames(topTags(de.tgw, n = sum(de.tgw$table$p.value
< 0.01) )$table)
```

s

```
> plotSmear(d, de.tags = detags.tgw, main = "FC plot using tagwise dispersions")
> abline(h = c(-2, 2), col = "dodgerblue")
```

In Figure 3, the 309 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 3 is very similar to that obtained using the common dispersion that we saw in Figure 2.

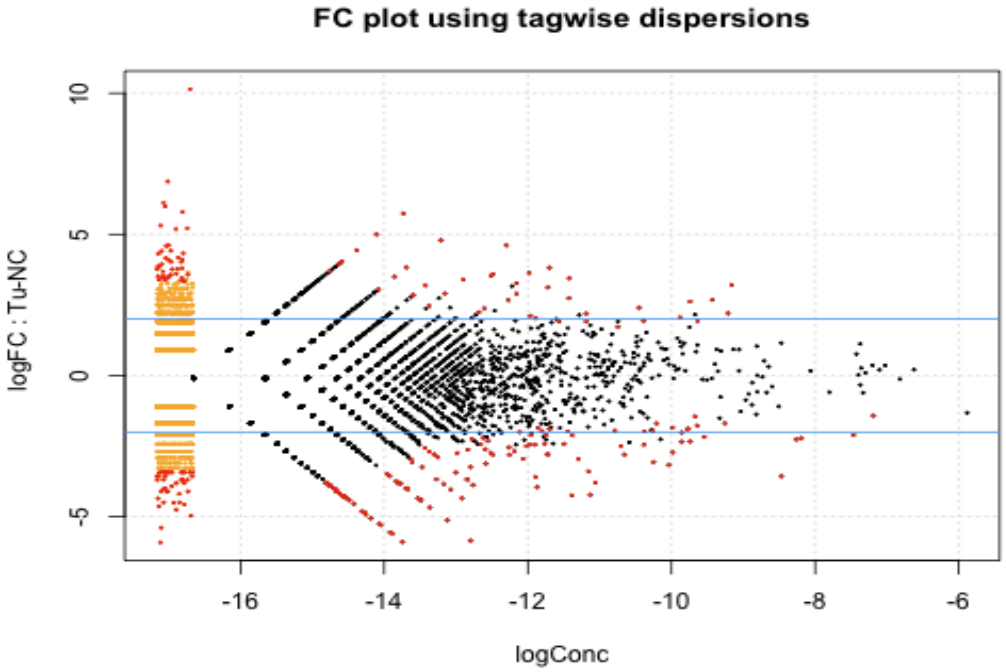

Figure 3: Plot of the log-fold change against the log-concentration for each tag. The 264 most differentially expressed tags as identified by edgeR are outlined in red.

8.6 Setup

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

```
> sessionInfo()

R version 2.9.1 (2009-06-26)
i386-apple-darwin8.11.1

locale:
```

```
en_AU.UTF-8/en_AU.UTF-8/C/C/en_AU.UTF-8/en_AU.UTF-8

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

other attached packages:
[1] edgeR_1.3.6

loaded via a namespace (and not attached):
[1] limma_2.18.2
```

and took 2–3 minutes to carry out on an Apple MacBook with a 2.4 Ghz Intel Core 2 Duo processor and 4 Gb of 1067 MHz DDR3 memory.

9 Case Study: deep-sequenced short tags

9.1 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].

9.2 Source of the data

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

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

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

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

The researchers obtained ~ 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.

9.3 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.

```
> setwd("/Users/davismcc/Documents/Honours/Data/Long_SAGE_Data")
> library(edgeR)
> 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
GSM272105 GSM272105.txt  DCLK transgenic (Dclk1) mouse hippocampus
GSM272106 GSM272106.txt    WT          wild-type mouse hippocampus
GSM272318 GSM272318.txt  DCLK transgenic (Dclk1) mouse hippocampus
GSM272319 GSM272319.txt    WT          wild-type mouse hippocampus
GSM272320 GSM272320.txt  DCLK transgenic (Dclk1) mouse hippocampus
GSM272321 GSM272321.txt    WT          wild-type mouse hippocampus
GSM272322 GSM272322.txt  DCLK transgenic (Dclk1) mouse hippocampus
GSM272323 GSM272323.txt    WT          wild-type mouse hippocampus
          lib.size
GSM272105  2685418
GSM272106  3517977
GSM272318  3202246
GSM272319  3558260
GSM272320  2460753
GSM272321   294909
GSM272322   651172
GSM272323  3142280

$counts
                  GSM272105 GSM272106 GSM272318 GSM272319 GSM272320
CATCGCCAGCGGGCACC         1         0         0         0         0
AAGGTCGACTCTGAAGT         1         1         0         0         0
CCTTCCTGGCTCTATGG         1         0         0         0         0
TCTGCTGAGCGTCTGTT         1         0         0         0         0
CCCCAGAGCGAATCAGG         1         1         2         1         1
                  GSM272321 GSM272322 GSM272323
CATCGCCAGCGGGCACC         0         0         0
AAGGTCGACTCTGAAGT         0         0         0
CCTTCCTGGCTCTATGG         0         0         0
TCTGCTGAGCGTCTGTT         0         0         0
CCCCAGAGCGAATCAGG         0         2         1
844311 more rows ...
```

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 with five or fewer counts in total—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.

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

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

9.4 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.

```
> plotMDS.dge(d, main="MDS Plot for 't Hoen Data", xlim=c(-2,1))
```

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 4. From this MDS plot we can identify sample GSM272322 as an outlier—there is a large distance between this sample and the others in Dimension 1. It would be a sensible approach to remove this sample from the dataset for analysis. In the analysis that follows, however, we do not remove the sample as we will just be demonstrating the use of edgeR.

9.5 Analysis using common dispersion

9.5.1 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`.

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

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

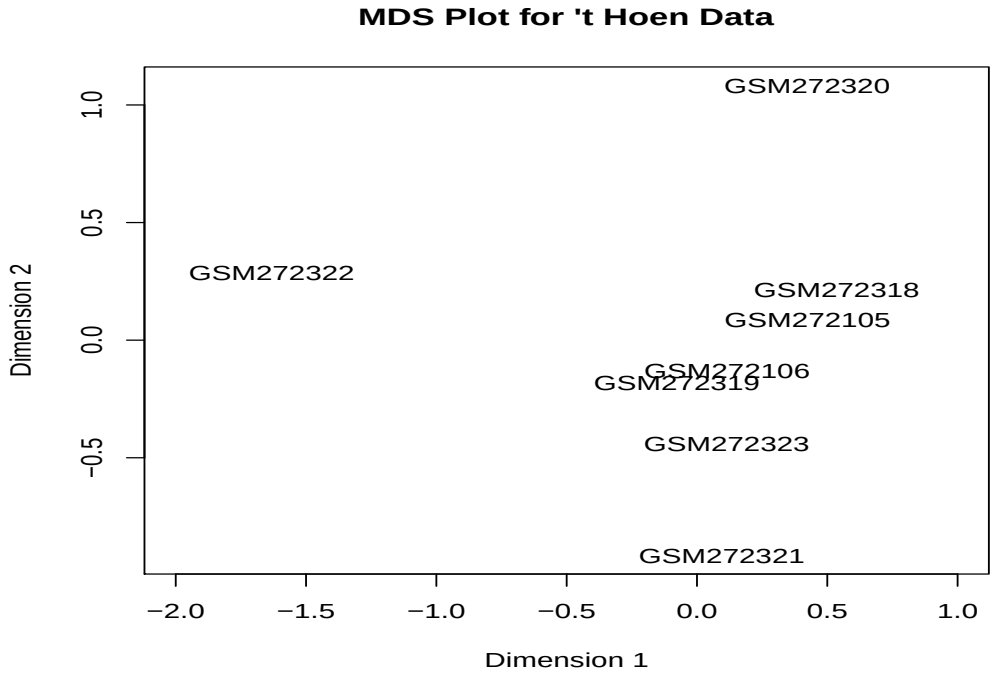

Figure 4: Multidimensional scaling (MDS) plot for the 't Hoen data, showing the relations between the samples in two dimensions. From this plot, the sample GSM272322 can be identified easily as an outlier—there is a large distance between this sample and the others in Dimension 1.

We see in the output below that the total counts in each library of the pseudocounts agrees well with the common library size, as desired.

```
> d$samples$lib.size

[1] 2685418 3517977 3202246 3558260 2460753  294909  651172 3142280

> d$common.lib.size

[1] 1885653

> colSums(d$pseudo.alt)

GSM272105 GSM272106 GSM272318 GSM272319 GSM272320 GSM272321 GSM272322
  1781297   1783960   1778180   1774107   1789066   1777588   1795899
GSM272323
  1783746
```

Here the coefficient of variation of biological variation (square root of the common dispersion) 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, 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] 0.1977094

> sqrt(d$common.dispersion)

[1] 0.4446452
```

9.5.2 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.common <- exactTest(d)

Comparison of groups:  WT - DCLK
```

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:  WT - DCLK
                     logConc     logFC       PValue          FDR
AATTTCTTCCTCTTCCT -17.29988 11.605093 1.028275e-36 1.149704e-31
CCGTCTTCTGCTTGTCG -10.57698  5.571165 1.076926e-23 6.020501e-19
TCTGTACGCAGTCAGGC -18.46570 -9.731983 7.100854e-23 2.646464e-18
CCGTCTTCTGCTTGTAA -14.44395  5.448158 1.083062e-21 3.027402e-17
CCGTCTTCTGCTTGTCA -15.45499  5.496920 2.759755e-20 6.171309e-16
```

```
AAGACTCAGGACTCATC -32.27026 35.491588 7.516087e-20 1.400610e-15
CCGTCTTCTGCTTGTAG -15.57138  4.803709 3.572708e-17 5.453064e-13
AGTGTGACGTGACCGGG -19.06213  8.067070 3.901700e-17 5.453064e-13
AAATTCTTCCTCTTCCT -19.14713  7.910596 2.838381e-16 3.526184e-12
TGTGTATCCCACAAGGG -18.68579  6.864501 5.262632e-16 5.884096e-12
```

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.

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

	GSM272105	GSM272106	GSM272318	GSM272319	GSM272320
AATTTCTTCCTCTTCCT	1	44	0	1	0
CCGTCTTCTGCTTGTCG	106	1485	268	420	601
TCTGTACGCAGTCAGGC	160	0	101	1	440
CCGTCTTCTGCTTGTAA	12	87	21	28	31
CCGTCTTCTGCTTGTCA	2	42	8	17	19
AAGACTCAGGACTCATC	0	6	0	2	0
CCGTCTTCTGCTTGTAG	9	61	11	20	17
AGTGTGACGTGACCGGG	0	249	0	2	1
AAATTCTTCCTCTTCCT	1	6	0	0	0
TGTGTATCCCACAAGGG	1	1	1	0	0

	GSM272321	GSM272322	GSM272323
AATTTCTTCCTCTTCCT	76	0	3487
CCGTCTTCTGCTTGTCG	5156	5	242
TCTGTACGCAGTCAGGC	0	33	0
CCGTCTTCTGCTTGTAA	352	1	14
CCGTCTTCTGCTTGTCA	183	1	17
AAGACTCAGGACTCATC	4	0	461
CCGTCTTCTGCTTGTAG	133	0	9
AGTGTGACGTGACCGGG	5	0	85
AAATTCTTCCTCTTCCT	2	0	288
TGTGTATCCCACAAGGG	6	0	252

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:  WT - DCLK
                    logConc     logFC       PValue          FDR
AAGACTCAGGACTCATC -32.27026  35.49159 7.516087e-20 1.400610e-15
CCTGATGCTACAGAAAA -32.72599  34.58013 5.104779e-15 5.188729e-11
CATAAGTCACAGAGTCG -32.76506 -34.50198 6.541738e-14 5.626348e-10
ACTCTGTGTATTACTCC -32.89435  34.24342 3.132417e-14 2.918603e-10
GATTTTTGTCGTGTTGG -32.95947  34.11317 2.185773e-13 1.745636e-09
AAAAGAAATCACAGTTG -32.96984 -34.09243 7.296827e-12 3.399379e-08
CACATAAGACTTTGGAC -33.09656  33.83899 2.181792e-12 1.434965e-08
AAAATGTTGTTTATGGA -33.10525  33.82162 4.049564e-12 2.156084e-08
GAAATTCTCCATTGATT -33.13607  33.75996 4.049564e-12 2.156084e-08
AAATTATTCCTCTTCCT -33.17108  33.68995 9.017868e-12 4.033115e-08
```

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 1710 tags (1.5% of the total number analysed) are significantly differentially expressed according to edgeR using the common dispersion estimate. Of those tags, 943 (55% of the DE tags) are up-regulated in the wild-type compared with the transgenic samples and 767 (45%) are down-regulated in the wild-type compared with transgenic mice.

```
> sum(p.adjust(de.common$table$p.value, method = "BH") < 0.085)

[1] 1710

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

[1] 1.529394

> top.com <- topTags(de.common, n = 1710)
> sum(top.com$table$logFC > 0)

[1] 943

> sum(top.com$table$logFC < 0)

[1] 767
```

9.5.3 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 5. In Figure 5, 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)

> plotSmear(d, de.tags = detags500.com, main = "FC plot using common dispersion")
> abline(h = c(-2, 2), col = "dodgerblue", lwd = 2)
```

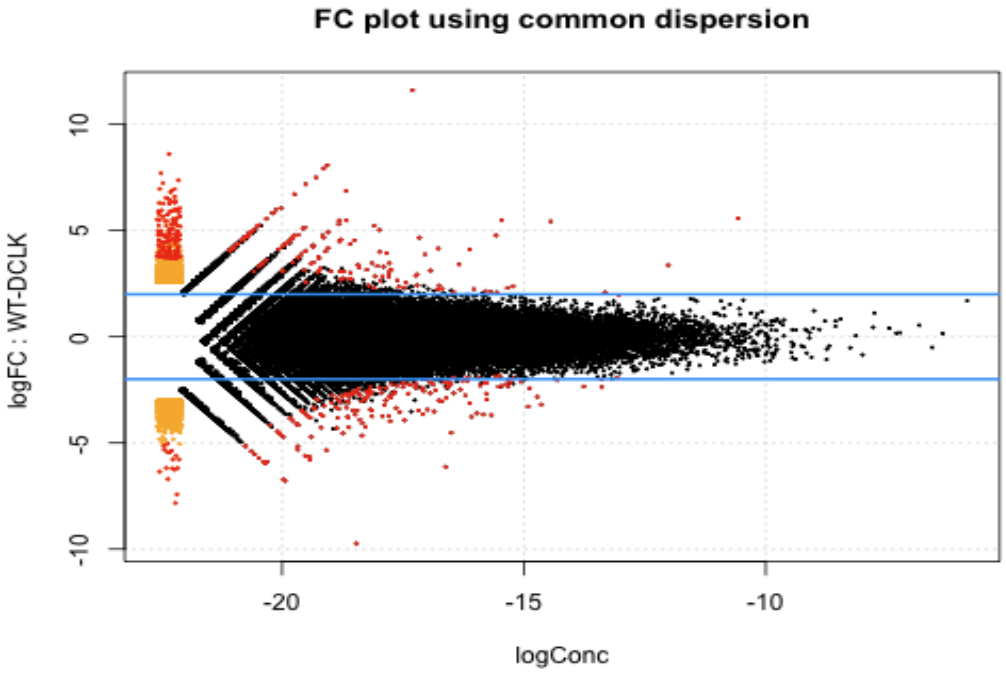

Figure 5: Plot of the log-fold change against the log-concentration for each tag. The 500 most differentially expressed tags as identified by edgeR using the common dispersion are outlined in red.

9.6 Analysis using moderated tagwise dispersions

9.6.1 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 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.

To run the moderated analysis, we need to determine how much moderation is necessary. As discussed in the SAGE case study, above, we currently recommend choosing a value for `prior.n` *a priori* that will provide an appropriate balance between the common and tagwise dispersion values. This moderation can improve the analysis by giving higher levels of significance to tags which have more consistent counts within groups (and therefore lower within-group variance) and reducing the significance of tags which have one extremely large count in one library, which can otherwise dominate the statistical assessment of differential expression.

In an experiment such as that we consider here, in which we have eight samples and thus six degrees of freedom for estimating the dispersion parameter, setting the `prior.n` to be ten should be appropriate. This means that the common likelihood receives the weight of ten individual tags, so there will be a reasonable degree of 'squeezing' towards the common dispersion estimate, but there is still enough scope to estimate an individual dispersion for each tag.

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.

```
> d <- estimateTagwiseDisp(d, prior.n = 10)

Using grid search to estimate tagwise dispersion.

> names(d)

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

> d$prior.n

[1] 10

> head(d$tagwise.dispersion)

[1] 0.1997564 0.1854905 0.1925808 0.1579529 0.2070189 0.1854905
```

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 0.09 to a maximum of 0.97, and the common dispersion estimate lies in between the median and mean values for the tagwise dispersion estimates.

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

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.08842 0.18550 0.19260 0.19930 0.20700 0.99000

> d$common.dispersion

[1] 0.1977094
```

9.6.2 Estimating the smoothing parameter using an approximate eBayes rule

Although we recommend using a fixed value for the `prior.n` parameter, chosen *a priori*, there is an alternative approach. Robinson and Smyth [2007] have developed an approximate empirical Bayes algorithm for estimating the smoothing parameter, and this method is implemented in edgeR. The smoothing parameter, `prior.n` can be calculated using the function `estimateSmoothing`, which takes the `DGEList` object produced by `estimateCommonDisp` as its argument. As we see below, for this dataset the estimate of the smoothing parameter is very small, so if we were to use this small value for the weight parameter we would moderate the tagwise dispersion estimates hardly at all. It would be equivalent to using the unmoderated tagwise dispersion estimates, which we do not trust as there are not enough samples to estimate the tagwise dispersions reliably. We will be much better off with greater moderation of the tagwise dispersions, as outlined above. At this time, it is recommended that the approximate empirical Bayes method for estimating the smoothing parameter be used as a guide, but not as a routine part of DGE analysis.

```
> prior.n <- estimateSmoothing(d)
> prior.n

[1] 0.0008078916
```

9.6.3 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)

Comparison of groups:  WT - DCLK
```

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:  WT - DCLK
                    logConc      logFC       PValue          FDR
TCTGTACGCAGTCAGGC -18.46778  -9.732317 2.271105e-18 2.539300e-13
CATAAGTCACAGAGTCG -32.76303 -34.506051 4.507171e-15 2.519711e-10
CCAAGAATCTGGTCGTA -17.46762  -3.929366 1.966857e-12 7.330412e-08
AATTTCTTCCTCTTCCT -17.30380  11.607564 3.139447e-12 8.775460e-08
ATACTGACATTTCGTAT -16.77800   4.144907 6.097301e-12 1.363466e-07
GCTAATAAATGGCAGAT -14.91149  -3.295240 9.212532e-12 1.716740e-07
CTGCTAAGCAGAAGCAA -16.99434  -3.427292 1.803488e-11 2.880660e-07
TTCCTGAAAATGTGAAG -17.08403  -3.639794 2.653336e-11 3.708335e-07
AAAAGAAATCACAGTTG -32.97330 -34.085504 6.032072e-11 7.493777e-07
AGTGTGACGTGACCGGG -19.07157   8.035868 1.581782e-10 1.768575e-06
```

Of course, we can also rank the top tags using the fold change instead of the p-value. The results of doing this are shown in the table below, but 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.tagwise, n = 10, sort.by = "logFC")

Comparison of groups:  WT - DCLK
                    logConc     logFC       PValue          FDR
AAGACTCAGGACTCATC -32.29053  35.45105 1.216554e-08 5.667572e-05
CCTGATGCTACAGAAAA -32.75498  34.52215 4.898918e-07 1.165411e-03
CATAAGTCACAGAGTCG -32.76303 -34.50605 4.507171e-15 2.519711e-10
ACTCTGTGTATTACTCC -32.92486  34.18238 2.898915e-08 1.200463e-04
AAAAGAAATCACAGTTG -32.97330 -34.08550 6.032072e-11 7.493777e-07
```

```
GATTTTTGTCGTGTTGG -33.00084  34.03043 1.747407e-07 4.765265e-04
CACATAAGACTTTGGAC -33.13007  33.77197 2.927280e-06 3.596662e-03
GAAATTCTCCATTGATT -33.15363  33.72485 7.690677e-07 1.500202e-03
AAAATGTTGTTTATGGA -33.15760  33.71690 5.539764e-06 5.567035e-03
AAATTATTCCTCTTCCT -33.18808  33.65595 1.494026e-06 2.456552e-03
```

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.

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

```
                  GSM272105 GSM272106 GSM272318 GSM272319 GSM272320
TCTGTACGCAGTCAGGC       160         0       101         1       440
CATAAGTCACAGAGTCG        67         0        77         0        58
CCAAGAATCTGGTCGTA        70         3        66         5        47
AATTTCTTCCTCTTCCT         1        44         0         1         0
ATACTGACATTTCGTAT         5       113         5       228         8
GCTAATAAATGGCAGAT       387        45       321        32       132
CTGCTAAGCAGAAGCAA        76         7        88         7        52
TTCCTGAAAATGTGAAG        74         6        70         9        86
AAAAGAAATCACAGTTG        31         0        90         0        42
AGTGTGACGTGACCGGG         0       249         0         2         1
                  GSM272321 GSM272322 GSM272323
TCTGTACGCAGTCAGGC         0        33         0
CATAAGTCACAGAGTCG         0         7         0
CCAAGAATCTGGTCGTA         0        13         7
AATTTCTTCCTCTTCCT        76         0      3487
ATACTGACATTTCGTAT         4         1       104
GCTAATAAATGGCAGAT         1        71        38
CTGCTAAGCAGAAGCAA         0        15        11
TTCCTGAAAATGTGAAG         0        10         7
AAAAGAAATCACAGTTG         0         3         0
AGTGTGACGTGACCGGG         5         0        85
```

```
> d$counts[detags.com, ]
```

	GSM272105	GSM272106	GSM272318	GSM272319	GSM272320
AATTTCTTCCTCTTCCT	1	44	0	1	0
CCGTCTTCTGCTTGTCG	106	1485	268	420	601
TCTGTACGCAGTCAGGC	160	0	101	1	440
CCGTCTTCTGCTTGTAA	12	87	21	28	31
CCGTCTTCTGCTTGTCA	2	42	8	17	19
AAGACTCAGGACTCATC	0	6	0	2	0
CCGTCTTCTGCTTGTAG	9	61	11	20	17
AGTGTGACGTGACCGGG	0	249	0	2	1
AAATTCTTCCTCTTCCT	1	6	0	0	0
TGTGTATCCCACAAGGG	1	1	1	0	0

	GSM272321	GSM272322	GSM272323
AATTTCTTCCTCTTCCT	76	0	3487
CCGTCTTCTGCTTGTCG	5156	5	242
TCTGTACGCAGTCAGGC	0	33	0
CCGTCTTCTGCTTGTAA	352	1	14
CCGTCTTCTGCTTGTCA	183	1	17
AAGACTCAGGACTCATC	4	0	461
CCGTCTTCTGCTTGTAG	133	0	9
AGTGTGACGTGACCGGG	5	0	85
AAATTCTTCCTCTTCCT	2	0	288
TGTGTATCCCACAAGGG	6	0	252

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 three 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] 3

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

[1] 76.7
```

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 1717 tags as DE using common dispersion and 1441 tags as DE using tagwise dispersions. This means that we determine 1.54% and 1.29% of tags to be DE using common and tagwise dispersions, respectively.

```
> sum(p.adjust(de.common$table$p.value, method = "BH") < 0.085)

[1] 1710

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

[1] 1.529394

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

[1] 1441

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

[1] 1.288805
```

Of the 1441 tags identified as DE using tagwise dispersions, 729 (51%) are up-regulated in wild-type and 712 (49%) 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 up-regulated in wild-type mice as DE.

```
> top.tgw <- topTags(de.tagwise, n = 1441)
> sum(top.tgw$table$logFC > 0)

[1] 729

> sum(top.tgw$table$logFC < 0)

[1] 712
```

9.6.4 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 6.

```
> detags500.com <- rownames(topTags(de.common, n = 500)$table)
> detags500.tgw <- rownames(topTags(de.tagwise, n = 500)$table)

> 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)
```

In Figure 6, 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 6, 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.

9.7 Setup

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

```
> sessionInfo()

R version 2.9.1 (2009-06-26)
i386-apple-darwin8.11.1

locale:
en_AU.UTF-8/en_AU.UTF-8/C/C/en_AU.UTF-8/en_AU.UTF-8

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

other attached packages:
[1] edgeR_1.3.6

loaded via a namespace (and not attached):
[1] limma_2.18.2
```

and took 5–10 minutes to carry out on an Apple MacBook with a 2.4 Ghz Intel Core 2 Duo processor and 4 Gb of 1067 MHz DDR3 memory.

10 Case Study: RNA-seq data

10.1 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.

10.2 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.

10.3 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.

[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.

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.

```
> setwd("/Users/davismcc/Documents/Honours/Data/LiData")
> library(edgeR)
> 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.

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

An object of class "DGEList"
$samples
        group lib.size
lane1 Control   978576
lane2 Control  1156844
lane3 Control  1442169
lane4 Control  1485604
lane5     DHT  1823460
lane6     DHT  1834335
lane8     DHT   681743

$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
ENSG00000124205     0     0     5     5     0     0     0
ENSG00000124207    76    80    85    97    80    81    37
21872 more rows ...
```

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

10.4 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 an MDS plot for the Li Data using the command below.

```
> plotMDS.dge(d.Li, main="MDS Plot for Li Data",xlim=c(-0.4,0.8))
```

The resulting plot for the Li data is shown in 7. In this plot, Lane 8 appears to be separated from the other samples in Dimension 1, but the distances between the samples are relatively small, so we need not treat this sample as an outlier and remove it from the analysis. Having now investigated some of the relationships between the samples we can proceed to the DE analysis of the data.

10.5 Analysis using common dispersion

10.5.1 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"        "conc"              "common.lib.size"

> d
```

```
An object of class "DGEList"
$samples
        group lib.size
lane1 Control   978576
lane2 Control  1156844
lane3 Control  1442169
lane4 Control  1485604
lane5     DHT  1823460
lane6     DHT  1834335
lane8     DHT   681743

$common.dispersion
[1] 0.02021596

$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
ENSG00000124205     0     0     5     5     0     0     0
ENSG00000124207    76    80    85    97    80    81    37
21872 more rows ...

$pseudo.alt
                      lane1        lane2      lane3      lane4     lane5
ENSG00000124208 623.7011345 682.94207324 555.660542 639.479771 336.71526
ENSG00000182463  34.3356362  22.18976707  23.832567  22.180621  33.16394
ENSG00000124201 235.0851553 240.62900197 259.559061 236.246548 262.61397
ENSG00000124205   0.1502441   0.05777303   4.539068   4.427231   0.00000
ENSG00000124207  98.4667235  88.25763482  75.005547  83.279817  55.73657
                    lane6        lane8
ENSG00000124208 499.57554 4.485689e+02
ENSG00000182463  38.29654 4.367427e+01
ENSG00000124201 209.30644 1.685512e+02
ENSG00000124205   0.00000 1.509887e-10
ENSG00000124207  56.12481 6.774033e+01
21872 more rows ...

$conc
$conc.common
ENSG00000124208 ENSG00000182463 ENSG00000124201 ENSG00000124205
   4.236152e-04    2.419602e-05    1.809021e-04    1.063407e-06
```

```
ENSG00000124207
   5.831078e-05
21872 more elements ...

$conc.group
                     Control          DHT
ENSG00000124208 4.897757e-04 3.350953e-04
ENSG00000182463 1.990269e-05 2.966863e-05
ENSG00000124201 1.902894e-04 1.684999e-04
ENSG00000124205 1.963536e-06 8.783496e-16
ENSG00000124207 6.735245e-05 4.640575e-05
21872 more rows ...


$common.lib.size
[1] 1276768
```

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.

We see in the output below that the total counts in each library of the pseudocounts agrees well with the common library size, as desired.

```
> d$samples$lib.size

[1]  978576 1156844 1442169 1485604 1823460 1834335  681743

> d$common.lib.size

[1] 1276768

> colSums(d$pseudo.alt)

  lane1   lane2   lane3   lane4   lane5   lane6   lane8
1277021 1276791 1276935 1277035 1277209 1277069 1277919
```

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 200 counts, then the estimate of the tag's variance is 5 times greater than it would be under the Poisson model.

```
> d$common.dispersion

[1] 0.02021596

> sqrt(d$common.dispersion)

[1] 0.1421828
```

10.5.2 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"
```

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 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        PValue           FDR
ENSG00000151503 -11.94799 5.705233 7.744047e-185 1.694165e-180
ENSG00000096060 -11.33288 4.893134 5.349073e-155 5.851083e-151
ENSG00000127954 -15.63280 8.118692 7.331162e-148 5.346128e-144
ENSG00000166451 -12.28742 4.570439 7.122773e-128 3.895623e-124
```

```
ENSG00000131016 -14.42856 5.190737 1.701099e-104 7.442990e-101
ENSG00000113594 -12.83343 4.000650  2.579172e-96  9.404091e-93
ENSG00000116285 -13.56732 4.087861  8.108902e-88  2.534264e-84
ENSG00000123983 -12.09539 3.544802  1.444749e-86  3.950846e-83
ENSG00000166086 -15.24551 5.390848  6.219612e-86  1.511849e-82
ENSG00000162772 -10.81704 3.201950  7.560702e-80  1.654055e-76
```

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)$table)
> d$counts[detags.com, ]
```

```
                lane1 lane2 lane3 lane4 lane5 lane6 lane8
ENSG00000151503    35    35    49    59  3307  3439  1224
ENSG00000096060    65    79   105   113  3975  3727  1451
ENSG00000127954     0     0     3     3   607   602   220
ENSG00000166451    41    52    57    57  1750  1654   728
ENSG00000131016     9     5    18     6   564   377   213
ENSG00000113594    37    36    57    43   936   959   418
ENSG00000116285    18    28    23    32   645   630   218
ENSG00000123983    62    76    94   108  1354  1258   628
ENSG00000166086     9     2     3     6   296   298   121
ENSG00000162772   172   204   250   304  2972  3269  1112
```

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       PValue          FDR
ENSG00000091972 -31.75811 -36.51589 8.689939e-56 6.789635e-53
ENSG00000164120 -32.27738  35.47735 4.636882e-44 2.205241e-41
ENSG00000100373 -32.93100 -34.17010 2.739292e-17 2.147939e-15
ENSG00000118513 -33.00607 -34.01998 8.893707e-16 5.895989e-14
ENSG00000081237 -33.15786 -33.71640 5.634630e-13 2.691459e-11
ENSG00000196660 -33.22302 -33.58606 3.870150e-12 1.647223e-10
ENSG00000117245 -33.23863 -33.55484 1.051358e-11 4.151725e-10
ENSG00000019549 -33.39510  33.24191 2.420329e-13 1.225684e-11
ENSG00000137404 -33.41447 -33.20316 1.041329e-08 2.556807e-07
ENSG00000059804 -33.43964  33.15284 2.174556e-12 9.630115e-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. Almost 5000 genes are given a p-value less than 0.01.

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

[1] 4760
```

The output below shows that over 4835 genes are given an adjusted p-value of less than 0.05. This means that if we set our control the FDR for differential expression at 5%, then edgeR identifies 22% of all the genes in the dataset as differentially expressed.

```
> sum(p.adjust(de.com$table$p.value, method = "BH") < 0.05)

[1] 4835

> mean(p.adjust(de.com$table$p.value, method = "BH") < 0.05) *
+     100

[1] 22.10084
```

Of the genes identified as DE above, 1911 (40% of the DE genes) are up-regulated in DHT-treated compared with control cells, and 2924 (60%) are up-regulated in the control cells compared with DHT-treated cells. It is interesting to note that although we detect far 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.

```
> top.com <- topTags(de.com, n = 4835)
> sum(top.com$table$logFC > 0)

[1] 1911

> sum(top.com$table$logFC < 0)

[1] 2924
```

10.5.3 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 8. In Figure 8, 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)

> plotSmear(d, de.tags = detags500.com, main = "FC plot using common dispersion")
> abline(h = c(-2, 2), col = "dodgerblue", lwd = 2)
```

10.6 Analysis using moderated tagwise dispersions

10.6.1 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.

As we only have seven libraries, a small sample size, we should not be too confident about the accuracy of the tagwise dispersions. Therefore it is recommended to use a larger value for `prior.n`, which can be selected *a priori*, instead of being estimated. In an experiment such as this, the seven samples mean that we have five degrees of freedom for estimating the dispersion parameter. Thus, setting the `prior.n` to be ten (as we have done previously) 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 estimatig the individual dispersion for each gene.

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.

```
> d <- estimateTagwiseDisp(d, prior.n = 10, 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] "conc"               "common.lib.size"
```

```
> d$prior.n
```

```
[1] 10
```

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

```
[1] 0.01936799 0.01729400 0.01936799 0.02145046 0.01522843 0.02774923
```

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 0.005 to a maximum of 0.236, and the common dispersion estimate lies in between the median and mean values for the tagwise dispersion estimates.

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

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.005025 0.019370 0.019370 0.020640 0.021450 0.236100
```

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

```
[1] 0.02021596
```

10.6.2 Estimating the smoothing parameter as an approximate eBayes rule

As discussed in the SAGE and Tag-seq case studies, we can use an approximate empirical Bayes rule that involves calculating a weight parameter prior.n. However, for many applications (especially if the estimated smoothing parameter is large), using the common dispersion for all tags will give excellent results. Also, if the estimated smoothing parameter is very small, then we prefer to select a larger smoothing parameter to squeeze the (in general unreliable) tagwise dispersion estimates towards the common value and thus stabilize these estimates. We currently recommend that the `prior.n` value given by `estimateSmoothing` be used as a guide, but not routinely in analysis.

The function `estimateSmoothing` estimates the smoothing parameter `prior.n` and takes the `DGEList` object produced by `estimateCommonDisp` as its argument. As we see below, for this dataset the estimate of the smoothing parameter is very small. If we were to use this tiny value for the weight parameter we would moderate the tagwise dispersion estimates so little that it would be equivalent to using just using the tagwise dispersion for each gene without any 'squeezing' at all towards the common dispersion. As the unmoderated tagwise dispersion estimates will be unreliable in experiments with small sample sizes, we recommend selecting a larger value for `prior.n`, as outlined above.

```
> prior.n <- estimateSmoothing(d)
> prior.n

[1] 0.002132353
```

10.6.3 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        PValue           FDR
ENSG00000151503 -11.947221 5.704024 2.214558e-301 4.844788e-297
ENSG00000096060 -11.332026 4.891305 1.545034e-259 1.690035e-255
ENSG00000166451 -12.288475 4.570589 8.545084e-180 6.231360e-176
ENSG00000127954 -15.632016 8.117608 5.274185e-171 2.884584e-167
ENSG00000162772 -10.816415 3.201287 6.821241e-136 2.984566e-132
ENSG00000113594 -12.834292 3.999954 8.573960e-119 3.126209e-115
ENSG00000116133 -11.741194 3.128433 2.367018e-118 7.397607e-115
ENSG00000116285 -13.566933 4.089714 4.303139e-116 1.176747e-112
ENSG00000115648  -8.831597 2.481756 1.967977e-114 4.783715e-111
ENSG00000130066 -10.322217 2.494893 6.742514e-108 1.475060e-104
```

Of course, we can also rank the top tags using the fold change instead of the p-value, as described above, 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.tgw, n = 10, sort.by = "logFC")
```

```
Comparison of groups:  DHT - Control
                  logConc     logFC       PValue          FDR
ENSG00000091972 -31.75810 -36.51592 2.059993e-63 1.502216e-60
ENSG00000164120 -32.27722  35.47767 1.732888e-44 5.123027e-42
ENSG00000100373 -32.93097 -34.17017 1.170463e-17 7.759461e-16
ENSG00000118513 -33.00748 -34.01715 6.961165e-15 3.696345e-13
ENSG00000081237 -33.15838 -33.71535 1.283161e-12 5.113246e-11
ENSG00000196660 -33.22317 -33.58577 3.427356e-12 1.288321e-10
ENSG00000117245 -33.23845 -33.55521 1.635293e-11 5.651708e-10
ENSG00000019549 -33.39438  33.24334 2.743940e-13 1.175645e-11
ENSG00000059804 -33.43993  33.15225 2.018339e-12 7.801273e-11
ENSG00000137404 -33.44594 -33.14022 5.612659e-07 9.183854e-06
```

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, but we do note that there is a great deal of consistency in the counts within groups for these top tags.

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

```
                lane1 lane2 lane3 lane4 lane5 lane6 lane8
ENSG00000151503    46    39    43    51  2315  2394  2293
ENSG00000096060    85    87    93    97  2783  2594  2717
ENSG00000166451    53    57    50    49  1225  1151  1361
ENSG00000127954     0     0     3     3   425   419   412
ENSG00000162772   225   225   221   262  2081  2276  2083
ENSG00000113594    48    40    51    37   655   667   780
ENSG00000116133   127   118   137   123  1143  1121  1041
ENSG00000116285    24    31    20    28   452   439   409
ENSG00000115648  1226  1196  1166  1156  6813  6920  6130
ENSG00000130066   403   426   440   410  2239  2301  2571
```

```
> round(d$pseudo.alt[detags.com, ])
```

	lane1	lane2	lane3	lane4	lane5	lane6	lane8
ENSG00000151503	46	39	43	51	2315	2394	2293
ENSG00000096060	85	87	93	97	2783	2594	2717
ENSG00000127954	0	0	3	3	425	419	412
ENSG00000166451	53	57	50	49	1225	1151	1361
ENSG00000131016	12	6	16	5	396	261	396
ENSG00000113594	48	40	51	37	655	667	780
ENSG00000116285	24	31	20	28	452	439	409
ENSG00000123983	81	84	83	93	948	875	1172
ENSG00000166086	11	2	3	5	207	207	226
ENSG00000162772	225	225	221	262	2081	2276	2083

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, 87% 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] 87.3
```

Using the common dispersion we found that 4835 genes (22% 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 more DE genes, namely 4933, or 23% of all of the genes in the dataset.

```
> sum(p.adjust(de.tgw$table$p.value, method = "BH") < 0.05)
```

```
[1] 4933
```

```
> mean(p.adjust(de.tgw$table$p.value, method = "BH") < 0.05) *
+     100
```

```
[1] 22.54880
```

Of the 4933 tags identified as DE using tagwise dispersions, 1981 (40%) are up-regulated in DHT-treated cells and 2952 (60%) 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 equal.

```
> top.tgw <- topTags(de.tgw, n = 4933)
> sum(top.tgw$table$logFC > 0)
```

```
[1] 1981
```

```
> sum(top.tgw$table$logFC < 0)
```

```
[1] 2952
```

10.6.4 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 9.

```
> detags500.com <- rownames(topTags(de.com, n = 500)$table)
> detags500.tgw <- rownames(topTags(de.tgw, n = 500)$table)

> 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)
```

In Figure 9, 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 9, 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.

10.7 Setup

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

```
> sessionInfo()

R version 2.9.1 (2009-06-26)
i386-apple-darwin8.11.1

locale:
en_AU.UTF-8/en_AU.UTF-8/C/C/en_AU.UTF-8/en_AU.UTF-8

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

other attached packages:
[1] edgeR_1.3.6

loaded via a namespace (and not attached):
[1] limma_2.18.2
```

and took 2–4 minutes to carry out on an Apple MacBook with a 2.4 Ghz Intel Core 2 Duo processor and 4 Gb of 1067 MHz DDR3 memory.

11 Case study: Oral carcinomas vs matched normal tissue

11.1 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.

11.2 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 data downloaded from GEO consisted of a file for each of the six samples, each of which contained mapped reads in AB's MAX format (similar to FASTA). The raw reads had been mapped by Tuch et al. [2010] to the UCSC hg18 reference genome. In order to analyse these data in R it was necessary to generate, from the MAX files, a file for each sample that contained the number of counts for each gene sequenced and successfully mapped, as well as the gene ID's corresponding to the reads.

AB supply some software tools for the analysis of RNA-seq data from the SOLiD platform. One function, `count_tags.pl` exists to count the number of reads mapping to each position of the genome and provide the associated gene and transcript ID's, as well as other information such as chromosome and precise start and end points of the read.

According to Tuch et al. [2010]'s paper, the reads were mapped to the UCSC hg18 reference genome. A reference annotation file was created using the AB function `refgene2gff.sh` (the default annotation for the AB software).

The file produced by `count_tags.pl` contains a row for each unique exon position in the genome to which reads map. For each row there are many columns which provide information including the matching gene and transcript ID's, the starting point and ending points for the reads and the count (number) of reads for that position in the genome. One such file was produced for each of the six samples.

The final task to produce data files that edgeR can handle was to simplify the files output by `count_tags.pl` into files with just one row for each gene ID and the appropriate count for each gene ID.

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

Our first task is to load the edgeR package, read the data into R and organise the data into a `DGEList` object that the functions in the package can recognise. The library size is usually the total sum of all of the counts for a library, and that is how library size is defined in this analysis. The easiest way to construct an appropriate `DGEList` object for these data is described below. In this case, the tag counts for the four individual libraries are stored in six separate plain text files.

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 experimental 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)
> setwd("C:/Users/yuchen/Documents/R/2010.05.15 Cox-Reid/Data/TuchData")
> targets <- read.delim(file="targets.txt",stringsAsFactors=FALSE,
+ row.names = "label")
> targets$tissue <- factor(targets$tissue)
> targets$patient <- factor(targets$patient)
> targets
```

```
                                files tissue patient
N8   GSM515513_N8_gene_agg_counts.txt normal       8
T8   GSM515514_T8_gene_agg_counts.txt tumour       8
N33 GSM515515_N33_gene_agg_counts.txt normal      33
T33 GSM515516_T33_gene_agg_counts.txt tumour      33
N51 GSM515517_N51_gene_agg_counts.txt normal      51
T51 GSM515518_T51_gene_agg_counts.txt tumour      51
```

```
> d.tuch <- readDGE(targets,skip=1,comment.char='#')
> colnames(d.tuch) <- row.names(targets)
> d.tuch <- calcNormFactors(d.tuch)
> d.tuch
```

```
An object of class "DGEList"
$samples
                                files tissue patient lib.size norm.factors
N8   GSM515513_N8_gene_agg_counts.txt normal       8  8859892       1.1511
T8   GSM515514_T8_gene_agg_counts.txt tumour       8  8270556       1.0695
N33 GSM515515_N33_gene_agg_counts.txt normal      33 17635449       0.7049
T33 GSM515516_T33_gene_agg_counts.txt tumour      33 15741712       0.9524
N51 GSM515517_N51_gene_agg_counts.txt normal      51 24009496       1.0427
T51 GSM515518_T51_gene_agg_counts.txt tumour      51 16967249       1.1603

$counts
         N8  T8   N33  T33   N51  T51
A1CF      0   0     0    0     0    0
A2BP1   188   1   154    1   711   57
```

```
A2LD1    10   0    18   13    40   22
A2M    2238 261  2375  612 15351 2003
A2ML1 11751 911 13305 3080  6935 1156
21783 more rows ...
```

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 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 20 000 unique tags sequenced, some 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 ten counts in total for a tag, we filter out tags with nine or fewer counts in total—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.

```
> d.tuch <- d.tuch[rowSums(d.tuch$counts) > 9, ]
```

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

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, as shown in Figure 10.

```
> plotMDS.dge(d.tuch, main="MDS Plot for Tuch Data")
```

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.

11.5 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.

```
> design <- model.matrix(~ patient + tissue, data = targets)
> design
```

```
    (Intercept) patient33 patient51 tissuetumour
N8            1         0         0            0
T8            1         0         0            1
N33           1         1         0            0
T33           1         1         0            1
N51           1         0         1            0
T51           1         0         1            1
attr(,"assign")
[1] 0 1 1 2
attr(,"contrasts")
attr(,"contrasts")$patient
[1] "contr.treatment"

attr(,"contrasts")$tissue
[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.

11.6 Analysis using Cox-Reid common dispersion

11.6.1 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 `estimateCRDisp()`. Once we have the design matrix, we pass it to the `estimateCRDisp()` function, together with the `DGEList` object 'd.tuch'.

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

[1] "samples"              "counts"
[3] "design"               "CR.common.dispersion"
```

The output of `estimateCRDisp` is a `DGEList` object with several new elements. The element `CR.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 0.2281, so the coefficient of biological variation is around 0.478.

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

[1] 0.2281214

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

[1] 0.4776205
```

11.6.2 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$CR.common.dispersion)
```

```
> lrt.tuch <- glmLRT(d.tuch, glmfit.tuch)
> options(digits = 4)
> topTags(lrt.tuch)
```

```
          logConc  logFC    PValue       FDR
TMPRSS11B  -12.49 -7.371 1.802e-20 3.926e-16
CKM        -11.66 -7.177 1.252e-18 1.364e-14
TNNC2      -14.60 -6.817 2.236e-18 1.624e-14
MAL        -13.11 -6.872 3.836e-18 2.089e-14
CRNN       -10.28 -7.247 5.259e-18 2.292e-14
PI16       -15.61 -6.574 2.457e-17 8.922e-14
MYBPC1     -11.96 -7.033 5.465e-17 1.701e-13
IL1F6      -14.74 -6.140 1.185e-16 3.228e-13
KRT36      -16.17 -7.342 2.215e-16 5.363e-13
MUC21      -12.89 -6.794 6.358e-16 1.385e-12
```

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
TMPRSS11B  2601  7874  3399  3  322    9
CKM        4120  5203 24175  5   24 1225
TNNC2       590  1627  1239  1    8   39
MAL        2742  3977  1772  3  264    8
CRNN      24178 22055 12533 49 2353   26
PI16        231   216  1950  0    2   35
MYBPC1     4791  4145 15766 10   14 1319
IL1F6       367  1825   809 10   45    1
KRT36       711   104    70  2    1    1
MUC21      4161  3432  1722  7  517    5
```

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")

       logConc  logFC    PValue       FDR
EMX1    -19.29  41.03 4.632e-10 5.607e-08
MAGEA4  -18.69  40.72 1.340e-09 1.448e-07
MAGEA3  -20.19  40.19 5.692e-08 3.626e-06
CR2     -19.89 -40.06 5.145e-08 3.336e-06
GSTA1   -19.85  40.01 1.873e-07 1.003e-05
PYY2    -20.06  39.98 1.617e-07 8.831e-06
GJB7    -20.63  39.96 9.991e-07 4.211e-05
CST1    -21.08  39.95 2.646e-07 1.356e-05
MAGEA6  -20.50  39.63 1.182e-06 4.825e-05
FBP2    -19.50 -39.36 4.798e-12 1.275e-09

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

       N8 N33 N51  T8 T33 T51
EMX1    0   0   0  43  67   0
MAGEA4  0   0   0 121   0  10
MAGEA3  0   0   0  38   0  14
CR2     0  73   8   0   0   0
GSTA1   0   0   0  51   9   0
PYY2    0   0   0  45   0   8
GJB7    0   0   0   0   0  73
CST1    0   0   0   0  10  40
MAGEA6  0   0   0  33   0   6
FBP2   40  38  28   0   0   0
```

We see in the output below that 2700 tags are significantly differentially expressed according to edgeR when using the CR common dispersion estimate and GLM likelihood ratio test. Of those tags, 530 are up-regulated in the tumour tissues compared with the normal tissues and 2170 are down-regulated in the tumour tissues compared with normal tissues.

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

   [,1]
-1  2170
0  19088
1    530
```

11.7 Analysis using Cox-Reid tagwise dispersion

11.7.1 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 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 thus 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. Here, we choose a moderate amount of smoothing—we let `prior.n` be 10. This means that the common likelihood receives the weight of 10 individual tags, so there will be a reasonable degree of 'squeezing', but there is still ample scope to estimate an individual dispersion for each tag.

The function `estimateCRDisp` produces a DGEList object that contains all of the CR tagwise dispersion estimates when the argument `'tagwise'` is set as `TRUE`, as we see below.

```
> d.tuch.tgw <- estimateCRDisp(d.tuch, design, tagwise = TRUE, prior.n = 10)
> names(d.tuch.tgw)

[1] "samples"                "counts"
[3] "design"                 "CR.common.dispersion"
[5] "CR.tagwise.dispersion"

> head(d.tuch.tgw$CR.tagwise.dispersion)

[1] 0.2281 0.2975 0.2357 0.2003 0.2108 0.2733
```

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 0.131

to a maximum of 0.777. 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.tgw$CR.tagwise.dispersion)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.131   0.206   0.224   0.229   0.230   0.777
```

11.7.2 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 ten. 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.tgw, design,
+ dispersion = d.tuch.tgw$CR.tagwise.dispersion)
> lrt.tuch.tgw <- glmLRT(d.tuch.tgw, 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)
```

```
          logConc  logFC    PValue       FDR
KRT36      -16.17 -7.347 9.574e-17 1.869e-12
PI16       -15.61 -6.595 1.716e-16 1.869e-12
TMPRSS11B  -12.49 -7.444 3.055e-15 2.218e-11
IL1F6      -14.74 -6.158 6.078e-15 2.693e-11
TNNC2      -14.60 -6.882 6.767e-15 2.693e-11
PTGFR      -15.38 -5.152 7.415e-15 2.693e-11
VTCN1      -18.81  7.691 8.146e-14 2.536e-10
DNAH17     -14.51  5.512 1.184e-13 2.948e-10
PYGM       -13.92 -5.532 1.218e-13 2.948e-10
PLA2G2A    -15.00 -5.587 2.304e-13 4.510e-10
```

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)$table)
> d.tuch.tgw$counts[top.tgw,order(d.tuch.tgw$samples$group)]
```

```
            N8  N33  N51  T8 T33  T51
KRT36      711  104   70   2   1    1
PI16       231  216 1950   0   2   35
TMPRSS11B 2601 7874 3399   3 322    9
IL1F6      367 1825  809  10  45    1
TNNC2      590 1627 1239   1   8   39
PTGFR      455  287 1736   7  12   46
VTCN1        0    1    0  29  81   84
DNAH17       4   22  107 678 614 2628
PYGM      1666 1420 2477  22  17  127
PLA2G2A    515  634 1914   1  29   52
```

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.tgw, sort.by = "logFC")
```

```
       logConc  logFC    PValue       FDR
EMX1    -19.29  41.03 4.442e-10 1.125e-07
MAGEA4  -18.69  40.72 1.321e-09 2.267e-07
MAGEA3  -20.19  40.19 5.553e-08 4.144e-06
CR2     -19.89 -40.06 5.096e-08 3.883e-06
GSTA1   -19.85  40.01 1.844e-07 1.126e-05
PYY2    -20.06  39.98 1.590e-07 9.897e-06
GJB7    -20.63  39.96 1.040e-06 4.903e-05
CST1    -21.08  39.95 2.602e-07 1.500e-05
MAGEA6  -20.50  39.63 1.166e-06 5.383e-05
FBP2    -19.50 -39.36 4.342e-12 4.300e-09
```

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

```
       N8 N33 N51  T8 T33 T51
EMX1    0   0   0  43  67   0
```

```
MAGEA4  0   0   0 121   0  10
MAGEA3  0   0   0  38   0  14
CR2     0  73   8   0   0   0
GSTA1   0   0   0  51   9   0
PYY2    0   0   0  45   0   8
GJB7    0   0   0   0   0  73
CST1    0   0   0   0  10  40
MAGEA6  0   0   0  33   0   6
FBP2   40  38  28   0   0   0
```

We see in the output below that 2626 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 2626 tags, 509 are up-regulated in the tumour tissues compared with the normal tissues and 2117 are down-regulated in the tumour tissues compared with normal tissues.

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

   [,1]
-1  2117
0  19162
1    509
```

11.8 Setup

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

```
> sessionInfo()

R version 2.12.0 alpha (2010-09-20 r52948)
Platform: i386-pc-mingw32/i386 (32-bit)

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

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

```
other attached packages:
[1] limma_3.5.20 MASS_7.3-8   edgeR_1.7.19

loaded via a namespace (and not attached):
[1] tools_2.12.0
```

12 Poisson example

It has been noted that, in some deep sequencing approaches, not a great deal of overdispersion is observed. Specifically, the means and variances appear to be very close to each other, suggesting the Poisson distribution is a good fit. Methods within the edgeR package may still be useful, including the quantile adjustment (effectively a normalization) and the exact testing routines.

To illustrate this, we sample Poisson data and carry out the exact testing procedure by setting the dispersion parameter in the NB model very close to zero. The NB model reduces to the Poisson model as the dispersion parameter goes to zero, and our investigations have shown that using the NB exact test with very small dispersion gives results entirely consistent with using a 'true' Poisson model and testing procedure. The data are quantile-adjusted before the exact test is carried out, with the dispersion parameter is set to (near) 0. The function `exactTest` operates only on `DGEList` objects, as illustrated in the case studies above, so we need to form a `DGEList` object containing our data before carrying out the Poisson test.

Nevertheless, an analysis using the Poisson distribution can be carried out as follows:

```
> library(edgeR)
> set.seed(101)
> n <- 10000
> lib.sizes <- c(40000, 50000, 38000, 40000)
> p <- runif(n, min = 1e-04, 0.001)
> mu <- outer(p, lib.sizes)
> mu[1:5, 3:4] <- mu[1:5, 3:4] * 8
> y <- matrix(rpois(4 * n, lambda = mu), nrow = n)
> dP <- DGEList(counts = y, group = rep(1:2, each = 2), lib.size = lib.sizes)
> dP$common.lib.size <- exp(mean(log(dP$samples$lib.size)))
```

And you can proceed as before:

```
> de.P <- exactTest(dP, dispersion = 1e-06)

Comparison of groups:  2 - 1

> topTags(de.P)
```

```
Comparison of groups: 2-1
            logConc     logFC       PValue          FDR
tag.3     -8.939897  2.946691 3.076540e-81 3.076540e-77
tag.4     -8.958118  2.910250 1.425021e-77 7.125106e-74
tag.1     -9.703306  2.937857 1.635095e-47 5.450317e-44
tag.5     -9.882605  2.579261 2.030845e-33 5.077113e-30
tag.2    -11.480315  3.310787 6.058052e-18 1.211610e-14
tag.9796 -11.366488  1.686444 3.397842e-06 5.663070e-03
tag.2893 -10.984522  1.132451 3.740766e-04 4.471809e-01
tag.142  -11.590674  1.394078 3.940818e-04 4.471809e-01
tag.3541 -13.561931  2.621489 4.024628e-04 4.471809e-01
tag.9783 -12.225718 -1.711087 6.061462e-04 6.061462e-01
```

13 Setup

This vignette was built on:

```
> sessionInfo()

R version 2.12.2 (2011-02-25)
Platform: i386-pc-mingw32/i386 (32-bit)

locale:
[1] LC_COLLATE=C
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

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

other attached packages:
[1] edgeR_2.0.5

loaded via a namespace (and not attached):
[1] limma_3.6.9  tools_2.12.2
```

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.

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%253Adoi%252F10.1371%252Fjournal.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.

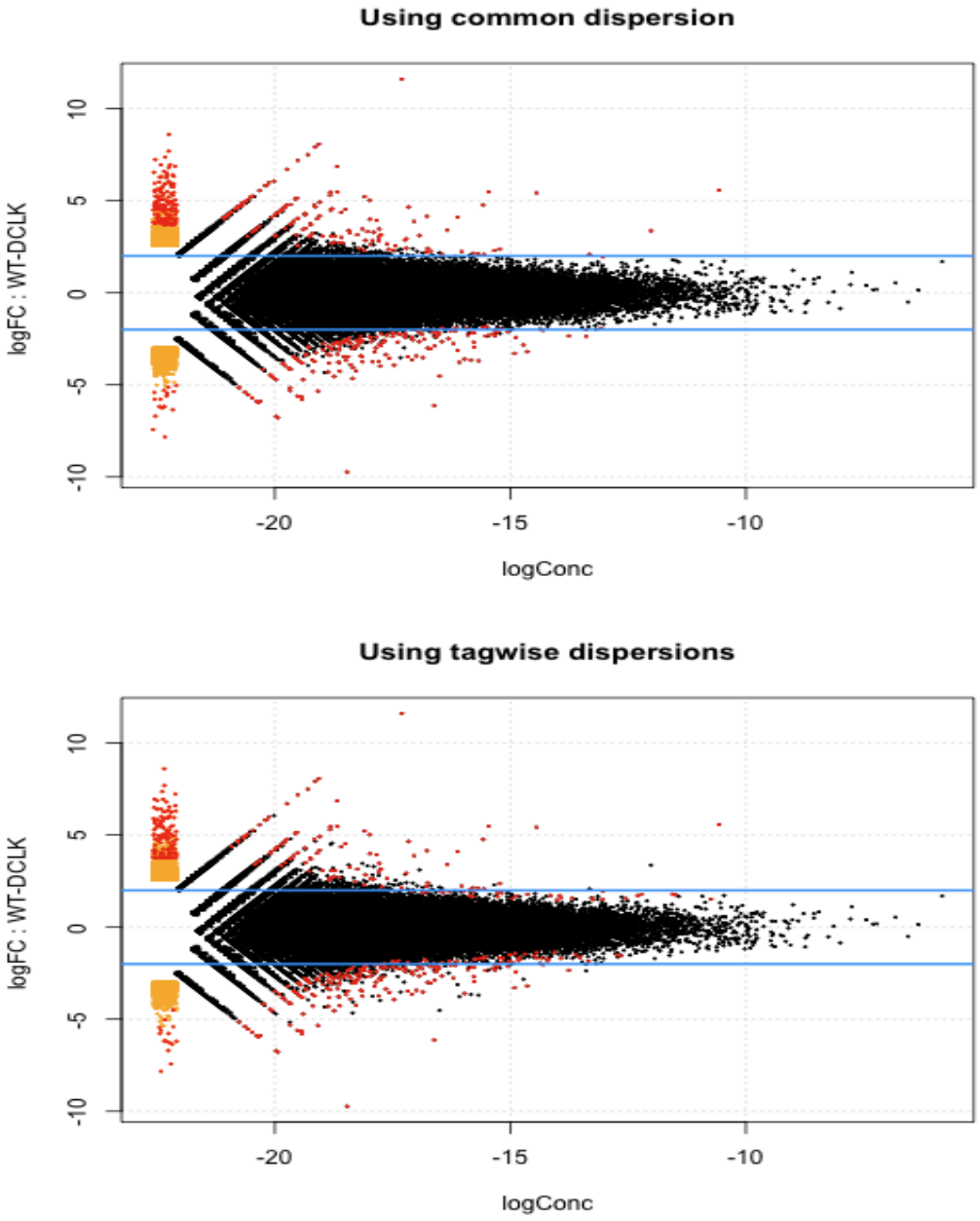

Figure 6: Plots of the log-fold change against the log-concentration for each tag, using the common dispersion (upper), and tagwise dispersions (lower). Tags with positive fold-change here are up-regulated in wild-type compared with transgenic mice. The 500 most differentially expressed tags according to each method are highlighted in red on both plots.

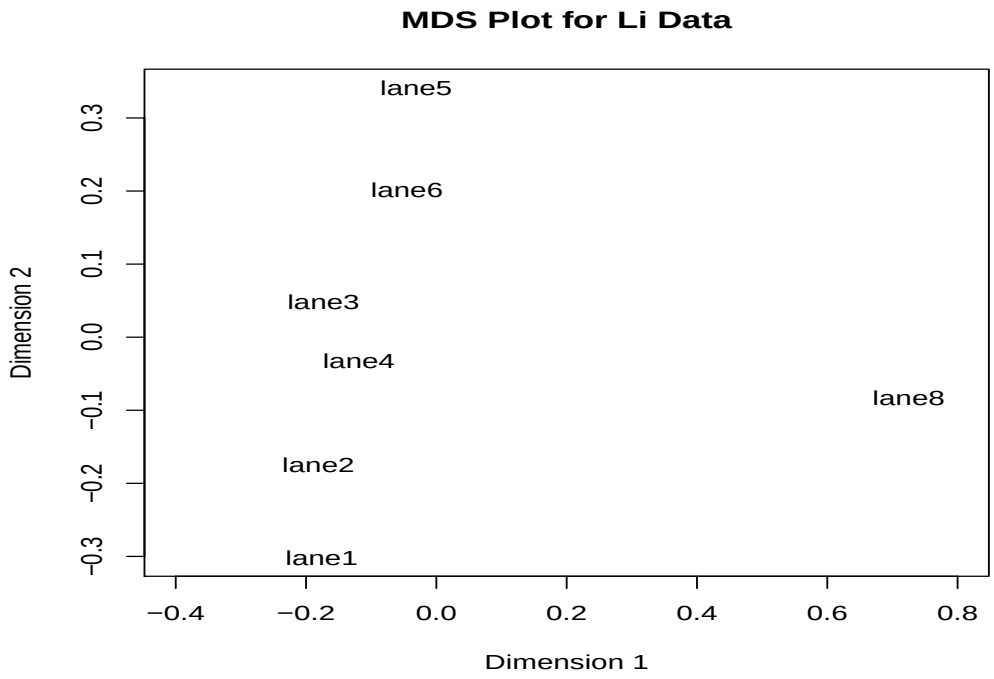

Figure 7: Multidimensional scaling (MDS) plot for the Li data, showin the relations between the samples in two dimensions. From this plot, Lane 8 may be identified easily as an outlier, but we note that the distance between this samples and the others in Dimension 1 is relatively small, so we do not remove this Lane 8 from the analysis.

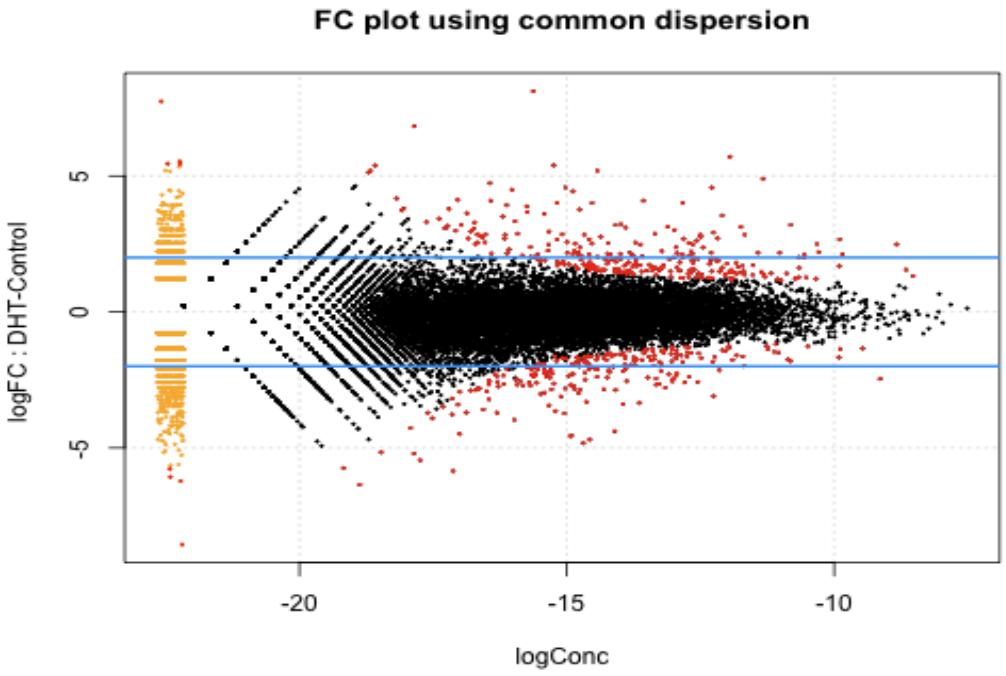

Figure 8: Plot of the log-fold change against the log-concentration for each tag. The 500 most differentially expressed tags as identified by edgeR using the common dispersion are outlined in red.

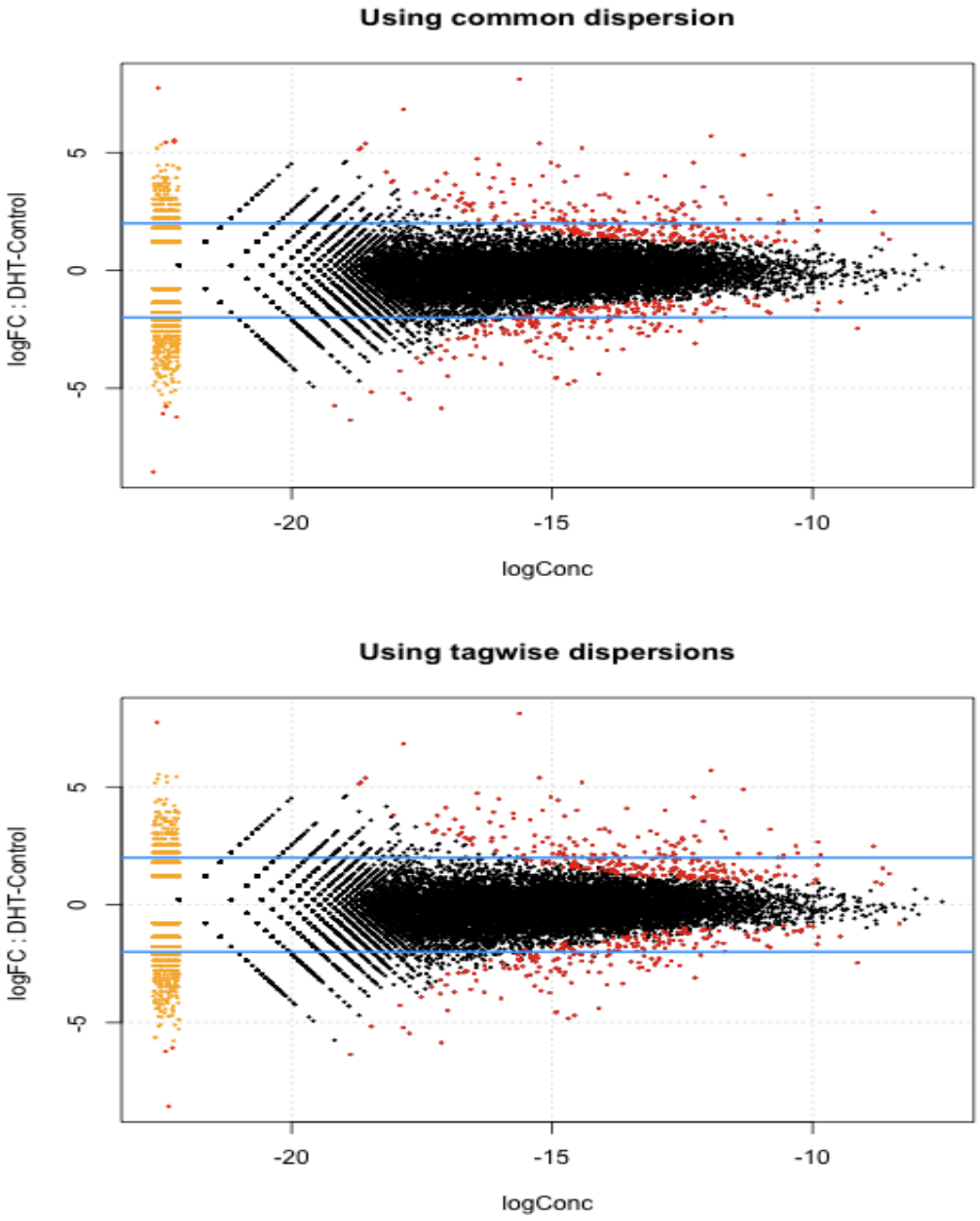

Figure 9: Plots of the log-fold change against the log-concentration for each tag, using the common dispersion (top), and tagwise dispersions (bottom). Tags with positive fold-change here are up-regulated in DHT-treated cells compared with control cells. The 500 most differentially expressed tags according to each method are highlighted in red on both plots.

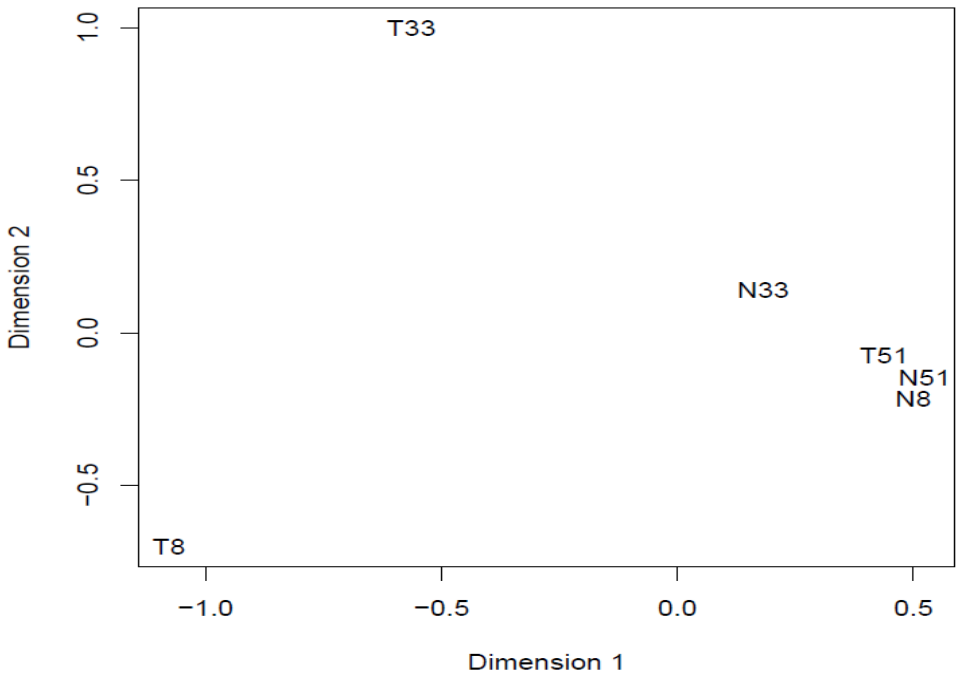

Figure 10: Multidimensional scaling (MDS) plot for the Tuch data, showing the relations between the samples in two dimensions. From this plot, the samples T33 and T8 can be identified easily as outliers—there is a large distance between these two samples and the others.