edgeR: differential analysis of sequence read count data

User's Guide

Yunshun Chen [1,2]***, Davis McCarthy*** [3,4]***, Pedro Baldoni*** [1,2]***, Matthew Ritchie*** [1,2]***, Mark Robinson*** [5]***, and Gordon Smyth*** [1,6]

[1]Walter and Eliza Hall Institute of Medical Research, Parkville, Victoria, Australia
[2]Department of Medical Biology, University of Melbourne, Victoria, Australia
[3]St Vincent's Institute of Medical Research, Fitzroy, Victoria, Australia
[4]Melbourne Integrative Genomics, University of Melbourne, Victoria, Australia
[5]Institute of Molecular Life Sciences and SIB Swiss Institute of Bioinformatics, University of Zurich, Zurich, Switzerland
[6]School of Mathematics and Statistics, University of Melbourne, Victoria, Australia

First edition 17 September 2008
Last revised 29 April 2024

Contents

Chapter 1

Introduction

1.1 Scope

This guide provides an overview of the Bioconductor package edgeR for differential expression analyses of read counts arising from RNA-Seq, SAGE or similar technologies [39]. The package can be applied to any technology that produces read counts for genomic features. Of particular interest are summaries of short reads from massively parallel sequencing technologies such as Illumina™, 454 or ABI SOLiD applied to RNA-Seq, SAGE-Seq or ChIP-Seq experiments, pooled shRNA-seq or CRISPR-Cas9 genetic screens and bisulfite sequencing for DNA methylation studies. edgeR provides statistical routines for assessing differential expression in RNA-Seq experiments or differential marking in ChIP-Seq experiments.

The package implements exact statistical methods for multigroup experiments developed by Robinson and Smyth [41, 42]. It also implements statistical methods based on generalized linear models (glms), suitable for multifactor experiments of any complexity, developed by McCarthy et al. [30], Lund et al. [28], Chen et al. [4] and Lun et al. [27]. Sometimes we refer to the former exact methods as *classic* edgeR, and the latter as *glm* edgeR. However the two sets of methods are complementary and can often be combined in the course of a data analysis. Most of the glm functions can be identified by the letters "`glm`" as part of the function name. The glm functions can test for differential expression using either likelihood ratio tests[30, 4] or quasi-likelihood F-tests [28, 27].

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

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

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

1.2 Citation

The edgeR package implements statistical methods from the following publications.

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

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

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

> Introduced empirical Bayes moderated dispersion parameter estimation. This is a crucial improvement on the previous idea of estimating the dispersions from a global model, because it permits gene-specific dispersion estimation to be reliable even for small samples. Gene-specific dispersion estimation is necessary so that genes that behave consistently across replicates should rank more highly than genes that do not.

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

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

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

> Introduced the idea of model-based library size normalization (aka "scale normalization") for RNA-Seq data. Proposed the TMM normalization method.

McCarthy, DJ, Chen, Y, Smyth, GK (2012). Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. *Nucleic Acids Research* 40, 4288-4297.

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

Lun, ATL, Chen, Y, and Smyth, GK (2016). It's DE-licious: a recipe for differential expression analyses of RNA-seq experiments using quasi-likelihood methods in edgeR. *Methods in Molecular Biology* 1418, 391–416.

> This book chapter explains the `glmQLFit` and `glmQLFTest` functions, which are alternatives to `glmFit` and `glmLRT`. They replace the chisquare approximation to the likelihood ratio statistic with a quasi-likelihood F-test, resulting in more conservative and rigorous type I error rate control.

Chen, Y, Lun, ATL, and Smyth, GK (2014). Differential expression analysis of complex RNA-seq experiments using edgeR. In: *Statistical Analysis of Next Generation Sequence Data*, Somnath Datta and Daniel S Nettleton (eds), Springer, New York.

> This book chapter explains the `estimateDisp` function and the weighted likelihood empirical Bayes method.

Zhou, X, Lindsay, H, and Robinson, MD (2014). Robustly detecting differential expression in RNA sequencing data using observation weights. *Nucleic Acids Research*, 42, e91.

Explains `estimateGLMRobustDisp`, which is designed to make the downstream tests done by `glmLRT` robust to outlier observations.

Dai, Z, Sheridan, JM, Gearing, LJ, Moore, DL, Su, S, Wormald, S, Wilcox, S, O'Connor, L, Dickins, RA, Blewitt, ME, and Ritchie, ME (2014). edgeR: a versatile tool for the analysis of shRNA-seq and CRISPR-Cas9 genetic screens. *F1000Research* 3, 95.

This paper explains the `processAmplicons` function for obtaining counts from the FASTQ files of shRNA-seq and CRISPR-Cas9 genetic screens and outlines a general workflow for analyzing data from such screens.

Chen, Y, Lun, ATL, and Smyth, GK (2016). From reads to genes to pathways: differential expression analysis of RNA-Seq experiments using Rsubread and the edgeR quasi-likelihood pipeline. *F1000Research* 5, 1438.

This paper describes a complete workflow of differential expression and pathway analysis using the edgeR quasi-likelihood pipeline.

Chen, Y, Pal, B, Visvader, JE, and Smyth, GK (2017). Differential methylation analysis of reduced representation bisulfite sequencing experiments using edgeR. *F1000Research* 6, 2055.

This paper explains a novel approach of detecting differentially methylated regions (DMRs) of reduced representation bisulfite sequencing (RRBS) experiments using edgeR.

Chen Y, Chen L, Lun ATL, Baldoni PL, Smyth GK (2024). edgeR 4.0: powerful differential analysis of sequencing data with expanded functionality and improved support for small counts and larger datasets. *bioRxiv* doi: 10.1101/2024.01.21.576131.

Summarizes all the capabilities of the edgeR package with emphasis on recently added functions.

1.3 How to get help

Most questions about edgeR will hopefully be answered by the documentation or references. If you've run into a question that isn't addressed by the documentation, or you've found a conflict between the documentation and what the software does, then there is an active support community that can offer help.

The edgeR authors always appreciate receiving reports of bugs in the package functions or in the documentation. The same goes for well-considered suggestions for improvements. All other questions or problems concerning edgeR should be posted to the Bioconductor support site https://support.bioconductor.org. Please send requests for general assistance and advice to the support site rather than to the individual authors. Posting questions to the Bioconductor support site has a number of advantages. First, the support site includes a community of experienced edgeR users who can answer most common questions. Second, the edgeR authors try hard to ensure that any user posting to Bioconductor receives assistance. Third, the support site allows others with the same sort of questions to gain from the answers. Users posting to the support site for the first time will find it helpful to read the posting guide at http://www.bioconductor.org/help/support/posting-guide.

The authors do not regularly answer questions posted to other forums, such as Biostars or SEQAnswers.

Note that each function in edgeR has its own online help page. For example, a detailed description of the arguments and output of the `estimateDisp` function can be read by typing `?estimateDisp` or `help("estimateDisp")` at the *R*prompt. If you have a question about any particular function, reading the function's help page will often answer the question very quickly. In any case, it is good etiquette to check the relevant help page first before posting a question to the support site.

1.4 Quick start

edgeR offers many variants on analyses. The glm approach is more popular than the classic approach as it offers great flexibilities. There are two testing methods under the glm framework: likelihood ratio tests and quasi-likelihood F-tests. The quasi-likelihood (QL) method is highly recommended for differential expression analyses of bulk RNA-seq data as it gives stricter error rate control by accounting for the uncertainty in dispersion estimation. The likelihood ratio test can be useful in some special cases such as single cell RNA-seq and datasets with no replicates. The details of these methods are described in Chapter 2.

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

```
> x <- read.delim("TableOfCounts.txt",row.names="Symbol")
> group <- factor(c(1,1,2,2))
> y <- DGEList(counts=x,group=group)
> keep <- filterByExpr(y)
> y <- y[keep,,keep.lib.sizes=FALSE]
> y <- normLibSizes(y)
> design <- model.matrix(~group)
> fit <- glmQLFit(y,design)
> qlf <- glmQLFTest(fit,coef=2)
> topTags(qlf)
```

1.5 Funding

The edgeR project has been supported by the National Health and Medical Research Council (Discovery grant 1050661) and by the Chan Zuckerberg Initiative (Essential Open Source Software for Science grants 2019-207283 and 2021-237445).

Chapter 2

Overview of capabilities

2.1 Terminology

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

2.2 Aligning reads to a genome

The first step in an RNA-seq analysis is usually to align the raw sequence reads to a reference genome, although there are many variations on this process. Alignment needs to allow for the fact that reads may span multiple exons which may align to well separated locations on the genome. We find the Subread-featureCounts pipeline [24, 25] to be very fast and effective for this purpose, but the STAR-featureCounts, STAR-htseq and Bowtie-TopHat-htseq pipelines are also popular. Subread and featureCounts are particularly convenient because it they implemented in the Bioconductor R package Rsubread [26].

2.3 Producing a table of read counts

edgeR works on a table of read counts, with rows corresponding to genes and columns to independent libraries. The counts represent the total number of reads aligning to each gene (or other genomic locus).

Such counts can be produced from aligned reads by a variety of short read software tools. We find the `featureCounts` function of the Rsubread package [25, 26] to be particularly effective and convenient, but other tools are available such as `findOverlaps` in the GenomicRanges package or the Python software `htseq-counts`.

Reads can be counted in a number of ways. When conducting gene-level analyses, the counts could be for reads mapping anywhere in the genomic span of the gene or the counts could be for exons only. We usually count reads that overlap any exon for the given gene, including the UTR as part of the first exon [25].

For data from pooled shRNA-seq or CRISPR-Cas9 genetic screens, the `processAmplicons` function [11] can be used to obtain counts directly from FASTQ files.

Note that edgeR is designed to work with actual read counts. We not recommend that predicted transcript abundances are input the edgeR in place of actual counts.

2.4 Reading the counts from a file

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

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

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

2.5 Pseudoalignment and quasi-mapping

The kallisto and Salmon software tools align sequence reads to the transcriptome instead of the reference genome and produce estimated counts per transcript. Output from either tool can be input to edgeR via the tximport package, which produces gene-level estimated counts and an associated edgeR offset matrix. Alternatively, kallisto or Salmon output can be read directly into edgeR using the `catchSalmon` and `catchKallisto` functions if the intention is to conduct a transcript-level analysis.

2.6 The DGEList data class

edgeR stores data in a simple list-based data object called a `DGEList`. This type of object is easy to use because it can be manipulated like any list in R. The function `readDGE` makes a `DGEList` object directly. If the table of counts is already available as a matrix or a data.frame, `x` say, then a `DGEList` object can be made by

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

A grouping factor can be added at the same time:

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

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

2.7 Filtering

Genes with very low counts across all libraries provide little evidence for differential expression. In the biological point of view, a gene must be expressed at some minimal level before it is likely to be translated into a protein or to be biologically important. In addition, the pronounced discreteness of these counts interferes with some of the statistical approximations that are used later in the pipeline. These genes should be filtered out prior to further analysis.

As a rule of thumb, genes are dropped if they can't possibly be expressed in all the samples for any of the conditions. Users can set their own definition of genes being expressed. Usually a gene is required to have a count of 5-10 in a library to be considered expressed in that library. Users should also filter with count-per-million (CPM) rather than filtering on the counts directly, as the latter does not account for differences in library sizes between samples.

Here is a simple example. Suppose the sample information of a `DGEList` object `y` is shown as follows:

```
> y$samples

        group lib.size norm.factors
Sample1     1 10880519            1
Sample2     1  9314747            1
Sample3     1 11959792            1
Sample4     2  7460595            1
Sample5     2  6714958            1
```

We filter out lowly expressed genes using the following commands:

```
> keep <- filterByExpr(y)
> y <- y[keep, , keep.lib.sizes=FALSE]
```

The `filterByExpr` function keeps rows that have worthwhile counts in a minumum number of samples (two samples in this case because the smallest group size is two). The function accesses the `group` factor contained in `y` in order to compute the minimum group size, but the filtering is performed independently of which sample belongs to which group so that no bias is introduced. It is recommended to recalculate the library sizes of the `DGEList` object after the filtering, although the downstream analysis is robust to whether this is done or not.

The `group` factor or the experimental design matrix can also be given directly to the `filterByExpr` function by

```
> keep <- filterByExpr(y, group=group)
```

if not already set in the `DGEList` object. More generally, `filterByExpr` can be used with any design matrix:

```
> keep <- filterByExpr(y, design)
```

In this form, the design matrix can be completely general, even including continuous covariates.

The filtering should be based on the grouping factors or treatment factors that will be involved in the differential expression teststested for, rather than on blocking variables that are not of scientific interest in themselves. For example, consider a paired comparison experiment in which the same treatment regimes applied to each of a number of subjects or patients:

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

In this design, `Patient` is included in the design matrix to correct for baseline differences between the Patients, but we will not be testing for differential expression between the Patients. The filtering should therefore be based soley `Treatment` rather than on `Patient`, i.e.,

```
> keep <- filterByExpr(y, group=Treatment)
```

rather than

```
> keep <- filterByExpr(y, design)
```

2.8 Normalization

2.8.1 Normalization is only necessary for sample-specific effects

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

2.8.2 Sequencing depth

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

2.8.3 Effective library sizes

The second most important technical influence on differential expression is one that is less obvious. RNA-seq provides a measure of the relative abundance of each gene in each RNA sample, but does not provide any measure of the total RNA output on a per-cell basis. In other words, RNA-seq measure relative expression rather than absolute expression. This becomes important for differential expression analyses when a small number of genes are very highly expressed in some samples but not in others. If a small proportion of highly expressed genes consume a substantial proportion of the total library size for a particular sample, this will cause the remaining genes to be under-sampled for that sample. Unless this effect is adjusted for, the remaining genes may falsely appear to be down-regulated in that sample [40].

The `normLibSizes` function normalizes the library sizes in such a way to minimize the log-fold changes between the samples for most genes. The default method for computing these scale factors uses a trimmed mean of M-values (TMM) between each pair of samples [40]. We call the product of the original library size and the scaling factor the *effective library size*, i.e., the normalized library size. The effective library size replaces the original library size in all downsteam analyses.

TMM is recommended for most RNA-Seq data where the majority (more than half) of the genes are believed not differentially expressed between any pair of the samples. If `y` is a DGEList object, then the following commands perform the TMM normalization and display the normalization factors.

```
> y <- normLibSizes(y)
> y$samples

        group lib.size norm.factors
Sample1     1 10880519         1.17
Sample2     1  9314747         0.86
Sample3     1 11959792         1.32
Sample4     2  7460595         0.91
Sample5     2  6714958         0.83
```

The set of all normalization factors for a DGEList multiply to unity, ensuring that the geometric mean of the effective library sizes is the same as the geometric mean of the original library sizes. A normalization factor below one indicates that a small number of high count genes are monopolizing the sequencing, causing the counts for other genes to be lower than would be usual given the library size. As a result, the library size will be scaled down, analogous to scaling the counts upwards in that library. Conversely, a factor above one scales up the library size, analogous to downscaling the counts.

2.8.4 GC content

The GC-content of each gene does not change from sample to sample, so it can be expected to have little effect on differential expression analyses to a first approximation. Recent publications, however, have demonstrated that sample-specific effects for GC-content can be detected [38, 20]. The EDASeq [38] and cqn [20] packages estimate correction factors that adjust for sample-specific GC-content effects in a way that is compatible with edgeR. In each case, the observation-specific correction factors can be input into the glm functions of edgeR as an *offset* matrix.

2.8.5 Gene length

Like GC-content, gene length does not change from sample to sample, so it can be expected to have little effect on differential expression analyses. Nevertheless, sample-specific effects for gene length have been detected [20], although the evidence is not as strong as for GC-content.

2.8.6 Model-based normalization, not transformation

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

Note that normalization in edgeR is model-based, and the original read counts are not themselves transformed. This means that users should not transform the read counts in any way before inputing them to edgeR. For example, users should not enter RPKM or FPKM values to edgeR in place of read counts. Such quantities will prevent edgeR from correctly estimating the mean-variance relationship in the data, which is a crucial to the statistical strategies underlying edgeR. Similarly, users should not add artificial values to the counts before inputing them to edgeR.

edgeR is not designed to work with estimated expression levels, for example as might be output by Cufflinks. edgeR can work with expected counts as output by RSEM, but raw counts are still preferred.

2.8.7 Pseudo-counts

The classic edgeR functions `estimateCommonDisp` and `exactTest` produce a matrix of *pseudo-counts* as part of the output object. The pseudo-counts are used internally to speed up computation of the conditional likelihood used for dispersion estimation and exact tests in the classic edgeR pipeline. The pseudo-counts represent the equivalent counts would have been observed had the library sizes all been equal, assuming the fitted model. The pseudo-counts are computed for a specific purpose, and their computation depends on the experimental design as well as the library sizes, so users are advised not to interpret the psuedo-counts as general-purpose normalized counts. They are intended mainly for internal use in the edgeR pipeline.

Disambiguation. Note that some other software packages use the term *pseudo-count* to mean something analogous to *prior counts* in edgeR, i.e., a starting value that is added to a zero count to avoid missing values when computing logarithms. In edgeR, a pseudo-count is a type of normalized count and a prior count is a starting value used to offset small counts.

2.9 Negative binomial models

2.9.1 Introduction

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

2.9.2 Biological coefficient of variation (BCV)

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

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

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

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

Dividing both sides by μ_{gi}^2 gives

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

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

Two levels of variation can be distinguished in any RNA-Seq experiment. First, the relative abundance of each gene will vary between RNA samples, due mainly to biological causes. Second, there is measurement error, the uncertainty with which the abundance of each gene in each sample is estimated by the sequencing technology. If aliquots of the same RNA sample are sequenced, then the read counts for a particular gene should vary according to a Poisson law [29]. If sequencing variation is Poisson, then it can be shown that the squared coefficient of variation (CV) of each count between biological replicate libraries is the sum of the squared CVs for technical and biological variation respectively,

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

Biological CV (BCV) is the coefficient of variation with which the (unknown) true abundance of the gene varies between replicate RNA samples. It represents the CV that would remain between biological replicates if sequencing depth could be increased indefinitely. The technical

CV decreases as the size of the counts increases. BCV on the other hand does not. BCV is therefore likely to be the dominant source of uncertainty for high-count genes, so reliable estimation of BCV is crucial for realistic assessment of differential expression in RNA-Seq experiments. If the abundance of each gene varies between replicate RNA samples in such a way that the genewise standard deviations are proportional to the genewise means, a commonly occurring property of measurements on physical quantities, then it is reasonable to suppose that BCV is approximately constant across genes. We allow however for the possibility that BCV might vary between genes and might also show a systematic trend with respect to gene expression or expected count.

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

2.9.3 Estimating BCVs

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

The parallel nature of sequencing data allows some possibilities for borrowing information from the ensemble of genes which can assist in inference about each gene individually. The easiest way to share information between genes is to assume that all genes have the same mean-variance relationship, in other words, the dispersion is the same for all the genes [42]. An extension to this "common dispersion" approach is to put a mean-dependent trend on a parameter in the variance function, so that all genes with the same expected count have the same variance.

However, the truth is that the gene expression levels have non-identical and dependent distribution between genes, which makes the above assumptions too naive. A more general approach that allows genewise variance functions with empirical Bayes moderation was introduced several years ago [41] and was extended to generalized linear models and thus more complex experimental designs [30]. Only when using tagwise dispersion will genes that are consistent between replicates be ranked more highly than genes that are not. It has been seen in many RNA-Seq datasets that allowing gene-specific dispersion is necessary in order that differential expression is not driven by outliers. Therefore, the tagwise dispersions are strongly recommended in model fitting and testing for differential expression.

In edgeR, we apply an empirical Bayes strategy for squeezing the tagwise dispersions towards a global dispersion trend or towards a common dispersion value. The amount of squeeze is determined by the weight given to the global value on one hand and the precision of the tagwise estimates on the other. The relative weights given to the two are determined the prior and residual degrees of freedom. By default, the prior degrees of freedom, which determines the amount of empirical Bayes moderation, is estimated by examining the heteroskedasticity of the data [4].

2.9.4 Quasi negative binomial

The NB model can be extended with quasi-likelihood (QL) methods to account for gene-specific variability from both biological and technical sources [28, 27]. Under the QL framework, the variance of the count y_{gi} is a quadratic function of the mean,

$$\text{var}(y_{gi}) = \sigma_g^2(\mu_{gi} + \phi\mu_{gi}^2),$$

where ϕ is the NB dispersion parameter and σ_g^2 is the QL dispersion parameter.

Any increase in the observed variance of y_{gi} will be modelled by an increase in the estimates for ϕ and/or σ_g^2. In this model, the NB dispersion ϕ is a global parameter whereas the QL is gene-specific, so the two dispersion parameters have different roles. The NB dispersion describes the overall biological variability across all genes. It represents the observed variation that is attributable to inherent variability in the biological system, in contrast to the Poisson variation from sequencing. The QL dispersion picks up any gene-specific variability above and below the overall level.

The common NB dispersion for the entire data set can be used for the global parameter. In practice, we use the trended dispersions to account for the empirical mean-variance relationships. Since the NB dispersion under the QL framework reflects the overall biological variability, it does not make sense to use the tagwise dispersions.

Estimation of the gene-specific QL dispersion is difficult as most RNA-seq data sets have limited numbers of replicates. This means that there is often little information to stably estimate the dispersion for each gene. To overcome this, an empirical Bayes (EB) approach is used whereby information is shared between genes [45, 28, 35]. Briefly, a mean-dependent trend is fitted to the raw QL dispersion estimates. The raw estimates are then squeezed towards this trend to obtain moderated EB estimates, which can be used in place of the raw values for downstream hypothesis testing. This EB strategy reduces the uncertainty of the estimates and improves testing power.

2.10 The classic edgeR pipeline: pairwise comparisons between two or more groups

2.10.1 Estimating dispersions

edgeR uses the quantile-adjusted conditional maximum likelihood (qCML) method for experiments with single factor.

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

The qCML method calculates the likelihood by conditioning on the total counts for each tag, and uses pseudo counts after adjusting for library sizes. Given a table of counts or a `DGEList` object, the qCML common dispersion and tagwise dispersions can be estimated using the `estimateDisp()` function. Alternatively, one can estimate the qCML common dispersion using the `estimateCommonDisp()` function, and then the qCML tagwise dispersions using the `estimateTagwiseDisp()` function.

However, the qCML method is only applicable on datasets with a single factor design since it fails to take into account the effects from multiple factors in a more complicated experiment. When an experiment has more than one factor involved, we need to seek a new way of estimating dispersions.

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

To estimate common dispersion and tagwise dispersions in one run (recommended):

```
> y <- estimateDisp(y)
```

Alternatively, to estimate common dispersion:

```
> y <- estimateCommonDisp(y)
```

Then to estimate tagwise dispersions:

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

Note that common dispersion needs to be estimated before estimating tagwise dispersions if they are estimated separately.

2.10.2 Testing for DE genes

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

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

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

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

2.11 More complex experiments (glm functionality)

2.11.1 Generalized linear models

Generalized linear models (GLMs) are an extension of classical linear models to nonnormally distributed response data [14]. GLMs specify probability distributions according to their mean-variance relationship, for example the quadratic mean-variance relationship specified above for read counts. Assuming that an estimate is available for ϕ_g, so the variance can be evaluated for any value of μ_{gi}, GLM theory can be used to fit a log-linear model

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

for each gene [30]. Here $\mathbf{x}_i$ is a vector of covariates that specifies the treatment conditions applied to RNA sample i, and $\boldsymbol{\beta}_g$ is a vector of regression coefficients by which the covariate effects are mediated for gene g. The quadratic variance function specifies the negative binomial GLM distributional family. The use of the negative binomial distribution is equivalent to treating the π_{gi} as gamma distributed.

2.11.2 Estimating dispersions

For general experiments (with multiple factors), edgeR uses the Cox-Reid profile-adjusted likelihood (CR) method in estimating dispersions [30]. 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 [8]. Given a table counts or a `DGEList` object and the design matrix of the experiment, generalized linear models are fitted. This allows valid estimation of the dispersion, since all systematic sources of variation are accounted for.

The CR method can be used to calculate a common dispersion for all the tags, trended dispersion depending on the tag abundance, or separate dispersions for individual tags. These can be done by calling the function `estimateDisp()` with a specified design. Alternatively, one can estimate the common, trended and tagwise dispersions separately using `estimateGLMCommonDisp()`, `estimateGLMTrendedDisp()` and `estimateGLMTagwiseDisp()`, respectively. The tagwise dispersion approach is strongly recommended in multi-factor experiment cases.

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

To estimate common dispersion, trended dispersions and tagwise dispersions in one run (recommended):

```
> y <- estimateDisp(y, design)
```

Alternatively, one can use the following calling sequence to estimate them one by one. To estimate common dispersion:

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

To estimate trended dispersions:

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

To estimate tagwise dispersions:

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

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

For more detailed examples, see the case study in Section 4.1 (Tuch's data), Section 4.2 (arabidopsis data), Section 4.3 (Nigerian data) and Section 4.4 (Fu's data).

2.11.3 Testing for DE genes

For general experiments, once dispersion estimates are obtained and negative binomial generalized linear models are fitted, we can proceed with testing procedures for determining differential expression using either quasi-likelihood (QL) F-test or likelihood ratio test.

While the likelihood ratio test is a more obvious choice for inferences with GLMs, the QL F-test is preferred as it reflects the uncertainty in estimating the dispersion for each gene. It provides more robust and reliable error rate control when the number of replicates is small. The QL dispersion estimation and hypothesis testing can be done by using the functions `glmQLFit()` and `glmQLFTest()`.

Given raw counts, NB dispersion(s) and a design matrix, `glmQLFit()` fits the negative binomial GLM for each tag and produces an object of class `DGEGLM` with some new components. This `DGEGLM` object can then be passed to `glmQLFTest()` to carry out the QL F-test. User can select one or more coefficients to drop from the full design matrix. This gives the null model against which the full model is compared. Tags can then be ranked in order of evidence for differential expression, based on the p-value computed for each tag.

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

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

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

To compare 2 vs 1:

```
> qlf.2vs1 <- glmQLFTest(fit, coef=2)
> topTags(qlf.2vs1)
```

To compare 3 vs 1:

```
> qlf.3vs1 <- glmQLFTest(fit, coef=3)
```

To compare 3 vs 2:

```
> qlf.3vs2 <- glmQLFTest(fit, contrast=c(0,-1,1))
```

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

To find genes different between any of the three groups:

```
> qlf <- glmQLFTest(fit, coef=2:3)
> topTags(qlf)
```

For more detailed examples, see the case study in Section 4.2 (arabidopsis data), Section 4.3 (Nigerian data) and Section 4.4 (Fu's data).

Alternatively, one can perform likelihood ratio test to test for differential expression. The testing can be done by using the functions `glmFit()` and `glmLRT()`. To apply the likelihood ratio test to the above example and compare 2 vs 1:

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

Similarly for the other comparisons.

For more detailed examples, see the case study in section 4.1 (Tuch's data)

2.12 What to do if you have no replicates

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

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

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

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

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

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

   In conjunction with this reduced design matrix, you could try `estimateGLMCommonDisp` with `method="deviance"`, `robust=TRUE` and `subset=NULL`. This is our current best attempt at an automatic method to estimate dispersion without replicates, although it will only

give good results when the counts are not too small and the DE genes are a small proportion of the whole. Please understand that this is only our best attempt to return something useable. Reliable estimation of dispersion generally requires replicates.

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

```
> y1 <- y
> y1$samples$group <- 1
```

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

```
> y0 <- estimateDisp(y1[housekeeping,], trend="none", tagwise=FALSE)
```

Then insert this into the full data object and proceed:

```
> y$common.dispersion <- y0$common.dispersion
> fit <- glmFit(y, design)
> lrt <- glmLRT(fit)
```

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

2.13 Differential expression above a fold-change threshold

All the above testing methods identify differential expression based on statistical significance regardless of how small the difference might be. On the other hand, one might be more interested in studying genes of which the expression levels change by a certain amount. A commonly used approach is to conduct DE tests, apply a fold-change cut-off and then rank all the genes above that fold-change threshold by p-value. In some other cases genes are first chosen according to a p-value cut-off and then sorted by their fold-changes. These combinations of p-value and fold-change threshold criteria seem to give more biological meaningful sets of genes than using either of them alone. However, they are both *ad hoc* and do not give meaningful p-values for testing differential expressions relative to a fold-change threshold. They favour lowly expressed but highly variable genes and destroy the control of FDR in general.

edgeR offers a rigorous statistical test for thresholded hypotheses under the GLM framework. It is analogous to TREAT [31] but much more powerful than the original TREAT method. Given a fold-change (or log-fold-change) threshold, the thresholded testing can be done by calling the function `glmTreat()` on a `DGEGLM` object produced by either `glmFit()` or `glmQLFit()`.

In the example shown in Section 2.11.3, suppose we are detecting genes of which the log2-fold-changes for 1 vs 2 are significantly greater than 1, i.e., fold-changes significantly greater than 2, we use the following commands:

```
> fit <- glmQLFit(y, design)
> tr <- glmTreat(fit, coef=2, lfc=1)
> topTags(tr)
```

Note that the fold-change threshold in `glmTreat()` is not the minimum value of the fold-change expected to see from the testing results. Genes will need to exceed this threshold by some way before being declared statistically significant. It is better to interpret the threshold as "the fold-change below which we are definitely not interested in the gene" rather than "the fold-change above which we are interested in the gene". In the presence of a huge number of DE genes, a relatively large fold-change threshold may be appropriate to narrow down the search to genes of interest. In the lack of DE genes, on the other hand, a small or even no fold-change threshold shall be used.

For more detailed examples, see the case study in Section 4.4 (Fu's data).

2.14 Gene ontology (GO) and pathway analysis

The gene ontology (GO) enrichment analysis and the KEGG pathway enrichment analysis are the common downstream procedures to interpret the differential expression results in a biological context. Given a set of genes that are up- or down-regulated under a certain contrast of interest, a GO (or pathway) enrichment analysis will find which GO terms (or pathways) are over- or under-represented using annotations for the genes in that set.

The GO analysis can be performed using the `goana()` function in edgeR. The KEGG pathway analysis can be performed using the `kegga()` function in edgeR. Both `goana()` and `kegga()` take a `DGELRT` or `DGEExact` object. They both use the NCBI RefSeq annotation. Therefore, the Entrez Gene identifier (ID) should be supplied for each gene as the row names of the input object. Also users should set `species` according to the organism being studied. The top set of most enriched GO terms can be viewed with the `topGO()` function, and the top set of most enriched KEGG pathways can be viewed with the `topKEGG()` function.

Suppose we want to identify GO terms and KEGG pathways that are over-represented in group 2 compared to group 1 from the previous example in Section 2.11.3 assuming the samples are collected from mice. We use the following commands:

```
> qlf <- glmQLFTest(fit, coef=2)
> go <- goana(qlf, species="Mm")
> topGO(go, sort="up")
> keg <- kegga(qlf, species="Mm")
> topKEGG(keg, sort="up")
```

For more detailed examples, see the case study in Section 4.1 (Tuch's data) and Section 4.4 (Fu's data).

2.15 Gene set testing

In addition to the GO and pathway analysis, edgeR offers different types of gene set tests for RNA-Seq data. These gene set tests are the extensions of the original gene set tests in limma in order to handle `DGEList` and `DGEGLM` objects.

The `roast()` function performs ROAST gene set tests [49]. It is a self-contained gene set test. Given a gene set, it tests whether the majority of the genes in the set are DE across the comparison of interest.

The `mroast()` function does ROAST tests for multiple sets, including adjustment for multiple testing.

The `fry()` function is a fast version of `mroast()`. It assumes all the genes in a set have equal variances. Since edgeR uses the z-score equivalents of NB random deviates for the gene set tests, the above assumption is always met. Hence, `fry()` is recommended over `roast()` and `mroast()` in edgeR. It gives the same result as `mroast()` with an infinite number of rotations.

The `camera()` function performs a competitive gene set test accounting for inter-gene correlation. It tests whether a set of genes is highly ranked relative to other genes in terms of differential expression [50].

The `romer()` function performs a gene set enrichment analysis. It implements a GSEA approach [47] based on rotation instead of permutation.

Unlike `goana()` and `kegga()`, the gene set tests are not limited to GO terms or KEGG pathways. Any pre-defined gene set can be used, for example MSigDB gene sets. A common application is to use a set of DE genes that was defined from an analysis of an independent data set.

For more detailed examples, see the case study in Section 4.3 (Nigerian's data) and Section 4.4 (Fu's data).

2.16 Clustering, heatmaps etc

The function `plotMDS` draws a multi-dimensional scaling plot of the RNA samples in which distances correspond to leading log-fold-changes between each pair of RNA samples. The leading log-fold-change is the average (root-mean-square) of the largest absolute log-fold-changes between each pair of samples. This plot can be viewed as a type of unsupervised clustering. The function also provides the option of computing distances in terms of BCV between each pair of samples instead of leading logFC.

Inputing RNA-seq counts to clustering or heatmap routines designed for microarray data is not straight-forward, and the best way to do this is still a matter of research. To draw a heatmap of individual RNA-seq samples, we suggest using moderated log-counts-per-million. This can be calculated by `cpm` with positive values for `prior.count`, for example

```
> logcpm <- cpm(y, log=TRUE)
```

where `y` is the normalized `DGEList` object. This produces a matrix of $\log_2$ counts-per-million (logCPM), with undefined values avoided and the poorly defined log-fold-changes for low counts shrunk towards zero. Larger values for `prior.count` produce stronger moderation of the values for low counts and more shrinkage of the corresponding log-fold-changes. The logCPM values can optionally be converted to RPKM or FPKM by subtracting $\log_2$ of gene length, see `rpkm()`.

2.17 Alternative splicing

edgeR can also be used to analyze RNA-Seq data at the exon level to detect differential splicing or isoform-specific differential expression. Alternative splicing events are detected by testing for differential exon usage for each gene, that is testing whether the log-fold-changes differ between exons for the same gene.

Both exon-level and gene-level tests can be performed simultaneously using the `diffSpliceDGE()` function in edgeR. The exon-level test tests for the significant difference between the exon's logFC and the overall logFC for the gene. Two testing methods at the gene-level are provided. The first is to conduct a gene-level statistical test using the exon-level test statistics. Whether it is a likelihood ratio test or a QL F-test depends on the pipeline chosen. The second is to convert the exon-level p-values into a genewise p-value by the Simes' method. The first method is likely to be powerful for genes in which several exons are differentially spliced. The Simes' method is likely to be more powerful when only a minority of the exons for a gene are differentially spliced.

The top set of most significant spliced genes can be viewed by the `topSpliceDGE()` function. The exon-level testing results for a gene of interest can be visualized by the `plotSpliceDGE()` function.

For more detailed examples, see the case study in Section 4.5 (Foxp1 data).

2.18 Differential transcript expression

edgeR offers an efficient way to assess differential transcript expression (DTE) [1]. This strategy is based on the outputs from the lightweight alignment tools kallisto [3] and Salmon [34]. In particular, the number of reads assigned to each transcript is quantified by kallisto and Salmon using a probabilistic approach. The read-to-transcript ambiguity is estimated using bootstrap samples from kallisto and Salmon and interpreted as technical overdispersions. These technical overdispersions can be incorporated into the downstream DTE analyses by a count-scaling approach. This allows DTE analyses to be conducted within the current edgeR framework exactly as for gene-level analyses.

To use edgeR for DTE analyses, users first need to run kallisto or Salmon for pseudo-alignment and quantification of reads at the transcript level with bootstrap resampling. Then the transcript-level counts can be imported and the technical overdispersions can be estimated using the `catchKallisto()` function or `catchSalmon()` function.

For more detailed examples, see the case study in Section 4.6 (human lung cell lines data).

2.19 CRISPR-Cas9 and shRNA-seq screen analysis

edgeR can also be used to analyze data from CRISPR-Cas9 and shRNA-seq genetic screens as described in Dai *et al.* (2014) [11]. Screens of this kind typically involve the comparison of two or more cell populations either in the presence or absence of a selective pressure, or as a time-course before and after a selective pressure is applied. The goal is to identify sgRNAs (or shRNAs) whose representation changes (either increases or decreases) suggesting that disrupting the target gene's function has an effect on the cell.

To begin, the `processAmplicons` function can be used to obtain counts for each sgRNA (or shRNA) in the screen in each sample and organise them in a `DGEList` for down-stream analysis using either the classic edgeR or GLM pipeline mentioned above. Next, gene set testing methods such as `camera` and `roast` can be used to summarize results from multiple sgRNAs or shRNAs targeting the same gene to obtain gene-level results.

For a detailed example, see the case study in Section 4.7 (CRISPR-Cas9 knockout screen analysis).

2.20 Bisulfite sequencing and differential methylation analysis

Cytosine methylation is a DNA modification generally associated with transcriptional silencing[43]. edgeR can be used to analyze DNA methylation data generated from bisulfite sequencing technology[6]. A DNA methylation study often involves comparing methylation levels at CpG loci between different experimental groups. Differential methylation analyses can be performed in edgeR for both whole genome bisulfite sequencing (WGBS) and reduced representation bisulfite sequencing (RRBS). This is done by considering the observed read counts of both methylated and unmethylated CpG's across all the samples. Extra coefficients are added to the design matrix to represent the methylation levels and the differences of the methylation levels betweeen groups.

See the case study in Section 4.8 (Bisulfite sequencing of mouse oocytes) for a detailed worked example of a differential methylation analysis. Another example workflow is given by Chen *et al* [6].

Chapter 3

Specific experimental designs

3.1 Introduction

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

Throughout this chapter we will assume that the read alignment, normalization and dispersion estimation steps described in the previous chapter have already been completed. We will assume that a `DGEList` object `y` has been created containing the read counts, library sizes, normalization factors and dispersion estimates.

3.2 Two or more groups

3.2.1 Introduction

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

```
> y$samples

        group lib.size norm.factors
Sample1     A   100001            1
Sample2     A   100002            1
Sample3     B   100003            1
Sample4     B   100004            1
Sample5     C   100005            1
```

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

```
> levels(y$samples$group)

[1] "A" "B" "C"
```

3.2.2 Classic approach

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

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

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

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

for C vs A, or

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

for B vs C.

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

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

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

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

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

Now

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

would still compare B to A, but

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

would now compare A to C.

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

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

compares B to A, whereas

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

compares A to C.

3.2.3 GLM approach

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

```
> design <- model.matrix(~0+group, data=y$samples)
> colnames(design) <- levels(y$samples$group)
> design
        A B C
Sample1 1 0 0
Sample2 1 0 0
Sample3 0 1 0
Sample4 0 1 0
Sample5 0 0 1
attr(,"assign")
[1] 1 1 1
attr(,"contrasts")
attr(,"contrasts")$group
[1] "contr.treatment"
```

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

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

```
> fit <- glmQLFit(y, design)
> qlf <- glmQLFTest(fit, contrast=c(-1,1,0))
> topTags(qlf)
```

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

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

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

One could make three pairwise comparisons between the groups by

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

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

Any comparison can be made. For example,

```
> qlf <- glmQLFTest(fit, contrast=c(-0.5,-0.5,1))
```

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

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

with the same results.

3.2.4 Questions and contrasts

The glm approach allows an infinite variety of contrasts to be tested between the groups. This embarassment of riches leads to the question, which specific contrasts should we test? This answer is that we should form and test those contrasts that correspond to the scientific questions that we want to answer. Each statistical test is an answer to a particular question, and we should make sure that our questions and answers match up.

To clarify this a little, we will consider a hypothetical experiment with four groups. The groups correspond to four different types of cells: white and smooth, white and furry, red and smooth and red furry. We will think of white and red as being the major group, and smooth and furry as being a sub-grouping. Suppose the RNA samples look like this:

Sample	Color	Type	Group
1	White	Smooth	A
2	White	Smooth	A
3	White	Furry	B
4	White	Furry	B
5	Red	Smooth	C
6	Red	Smooth	C
7	Red	Furry	D
8	Red	Furry	D

To decide which contrasts should be made between the four groups, we need to be clear what are our scientific hypotheses. In other words, what are we seeking to show?

First, suppose that we wish to find genes that are always higher in red cells than in white cells. Then we will need to form the four contrasts `C-A`, `C-B`, `D-A` and `D-B`, and select genes that are significantly up for all four contrasts.

Or suppose we wish to establish that the difference between Red and White is large compared to the differences between Furry and Smooth. An efficient way to establish this would be to form the three contrasts `B-A`, `D-C` and `(C+D)/2-(A+B)/2`. We could confidently make this assertion for genes for which the third contrast is far more significant than the first two. Even if `B-A` and `D-C` are statistically significant, we could still look for genes for which the fold changes for `(C+D)/2-(A+B)/2` are much larger than those for `B-A` or `D-C`.

We might want to find genes that are more highly expressed in Furry cells regardless of color. Then we would test the contrasts `B-A` and `D-C`, and look for genes that are significantly up for both contrasts.

Or we want to assert that the difference between Furry over Smooth is much the same regardless of color. In that case you need to show that the contrast `(B+D)/2-(A+C)/2` (the average Furry effect) is significant for many genes but that `(D-C)-(B-A)` (the interaction) is not.

3.2.5 A more traditional glm approach

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

```
> design <- model.matrix(~group, data=y$samples)
> design

        (Intercept) groupB groupC
Sample1           1      0      0
Sample2           1      0      0
Sample3           1      1      0
Sample4           1      1      0
Sample5           1      0      1
attr(,"assign")
[1] 0 1 1
attr(,"contrasts")
attr(,"contrasts")$group
[1] "contr.treatment"
```

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

```
> fit <- glmQLFit(y, design)
> qlf <- glmQLFTest(fit, coef=2)
```

would test for differential expression in B vs A. and

```
> qlf <- glmQLFTest(fit, coef=3)
```

would test for differential expression in C vs A.

This parametrization makes good sense when the first group represents a reference or control group, as all comparison are made with respect to this condition. If we releveled the factor to make level C the first level (see Section 3.2.2), then the design matrix becomes:

```
> design2 <- model.matrix(~group, data=y2$samples)
> design2

        (Intercept) groupA groupB
Sample1           1      1      0
Sample2           1      1      0
Sample3           1      0      1
Sample4           1      0      1
```

```
Sample5           1      0      0
attr(,"assign")
[1] 0 1 1
attr(,"contrasts")
attr(,"contrasts")$group
[1] "contr.treatment"
```

Now

```
> fit2 <- glmQLFit(y, design2)
> qlf <- glmQLFTest(fit2, coef=2)
```

compares A to C, and

```
> qlf <- glmQLFTest(fit2, coef=3)
```

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

```
> qlf <- glmQLFTest(fit2, contrast=c(0,-1,1))
```

Note that

```
> qlf <- glmQLFTest(fit2, coef=1)
```

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

3.2.6 An ANOVA-like test for any differences

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

```
> qlf <- glmQLFTest(fit, coef=2:3)
> topTags(qlf)
```

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

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

```
> qlf <- glmQLFTest(fit2, coef=2:3)
```

gives exactly the same results, even though `fit2` and `fit` were computed using different design matrices. Here `fit2` is as defined in the previous section.

3.3 Experiments with all combinations of multiple factors

3.3.1 Defining each treatment combination as a group

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

```
> targets

           Treat Time
Sample1  Placebo   0h
Sample2  Placebo   0h
Sample3  Placebo   1h
Sample4  Placebo   1h
Sample5  Placebo   2h
Sample6  Placebo   2h
Sample7     Drug   0h
Sample8     Drug   0h
Sample9     Drug   1h
Sample10    Drug   1h
Sample11    Drug   2h
Sample12    Drug   2h
```

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

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

           Treat Time      Group
Sample1  Placebo   0h Placebo.0h
Sample2  Placebo   0h Placebo.0h
Sample3  Placebo   1h Placebo.1h
Sample4  Placebo   1h Placebo.1h
Sample5  Placebo   2h Placebo.2h
Sample6  Placebo   2h Placebo.2h
Sample7     Drug   0h    Drug.0h
Sample8     Drug   0h    Drug.0h
Sample9     Drug   1h    Drug.1h
Sample10    Drug   1h    Drug.1h
Sample11    Drug   2h    Drug.2h
Sample12    Drug   2h    Drug.2h
```

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

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

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

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

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

To find genes responding to the drug at 1 hour:

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

or at 2 hours:

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

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

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

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

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

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

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

so

```
> qlf <- glmQLFTest(fit, contrast=c(-1,0,1,0,0,0))
```

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

```
> qlf <- glmQLFTest(fit, contrast=c(-1,0,1,1,0,-1))
```

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

3.3.2 Nested interaction formulas

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

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

Then form the design matrix:

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

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

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

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

```
> qlf <- glmQLFTest(fit, coef=2)
```

is the baseline drug vs placebo comparison,

```
> qlf <- glmQLFTest(fit, coef=4)
```

is the drug effect at 1 hour,

```
> qlf <- glmQLFTest(fit, coef=6)
```

is the drug effect at 2 hours, and finally

```
> qlf <- glmQLFTest(fit, contrast=c(0,0,0,0-1,1))
```

is the `DrugvsPlacebo.2h` contrast.

3.3.3 Treatment effects over all times

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

```
> qlf <- glmQLFTest(fit, coef=c(4,6))
```

finds genes that respond to the treatment at either 1 hour or 2 hours versus the 0 hour baseline. This is analogous to an ANOVA F-test for a normal linear model.

3.3.4 Interaction at any time

A very traditional approach taken in many statistics textbooks would be to specify our experiment in terms of a factorial model:

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

which is equivalent to

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

While the factorial model has a long history in statistics, the coefficients are more difficult to interpret than for the design matrices in Sections 3.3.1 or 3.3.2 and the coefficients are generally less biologically meaningful.

In the factorial model, the coefficient names are:

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

Now

```
> qlf <- glmQLFTest(fit, coef=2)
```

is again the baseline drug vs placebo comparison at 0 hours, but

```
> qlf <- glmQLFTest(fit, coef=3)
```

and

```
> qlf <- glmQLFTest(fit, coef=4)
```

are the effects of the reference drug, i.e., the effects of the placebo at 1 hour and 2 hours. In most experimental studies, none of the above three tests are would be of any particular scientific interest.

The factorial formula is primarily useful as a way to conduct an overall test for interaction. The last two coefficients correspond to the interaction contrasts `(Drug.1h-Placebo.1h)-(Drug.0h-Placebo.0h)` and `(Drug.2h-Placebo.2h)-(Drug.0h-Placebo.0h)` respectively, which are the same as the contrasts `DrugvsPlacebo.1h` and `DrugvsPlacbo.2h` defined in Section 3.3.1. Hence

```
> qlf <- glmQLFTest(fit, coef=5:6)
```

is useful because it detects genes that respond *differently* to the drug, relative to the placebo, at either of the times. In other words, specifying `coef=5:6` in the GLM test is a way to test for interaction between treatment without having to form the interaction contrasts explicitly. The results will be the same as if we had specified

```
> qlf <- glmQLFTest(fit, contrast=my.contrasts[,"DrugvsPlacebo.1h","DrugvsPlacebo.2h"])
```

in Section 3.3.1.

3.4 Additive models and blocking

3.4.1 Paired samples

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

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

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

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

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

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

As always, the dispersion has to be estimated:

```
> y <- estimateDisp(y,design)
```

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

```
> fit <- glmQLFit(y, design)
> qlf <- glmQLFTest(fit)
> topTags(qlf)
```

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

See the oral carcinomas case study of Section 4.1 for a fully worked analysis with paired samples.

3.4.2 Blocking

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

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

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

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

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

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

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

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

```
> qlf <- glmQLFTest(fit, coef=4:5)
> topTags(qlf)
```

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

```
> qlf <- glmQLFTest(fit, coef=4)
> topTags(qlf)
```

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

```
> qlf <- glmQLFTest(fit, coef=5)
> topTags(qlf)
```

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

```
> qlf <- glmQLFTest(fit, contrast=c(0,0,0,-1,1))
> topTags(qlf)
```

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

3.4.3 Batch effects

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

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

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

The Arabidopsis case study in Section 4.2 gives a fully worked example with batch effects.

3.5 Comparisons both between and within subjects

Here is a more complex scenario, posed by a poster to the Bioconductor mailing list. The experiment has 18 RNA samples collected from 9 subjects. The samples correspond to 3 healthy patients, 3 patients with Disease 1 and 3 patients with Disease 2. Each patient received two treatments, an active treatment with a hormone and a control treatment without the hormone. The targets frame looks like this:

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

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

This type of experiment is sometimes called a *multilevel design* or a *repeated measures* design. The *repeated measures* refer to the multiple measurements made on each patient. The experiment is *multilevel* in the sense that there are two error levels, one between patients and one for the repeat measurements within patients. The repeated measurements made on each patient will typically be more similar to each other than measurements made on different patients. Hence the variability within patients is typically lower than that between patients.

The easiest way to approach this design is to view it as three paired-comparison experiments, one for each disease group. We can compare the hormone treatment to the control treatment separately for each disease group, then contrast how effect the hormone is between the three disease groups.

For we define the experimental factors:

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

We need to adjust for baseline differences between the patients, so the first step is to initialize the design matrix with patient effects:

```
> design <- model.matrix(~Patient)
```

Then we define disease-specific treatment effects and append them to the design matrix:

```
> Healthy.Hormone <- Disease== "Healthy" & Treatment=="Hormone"
> Disease1.Hormone <- Disease=="Disease1" & Treatment=="Hormone"
> Disease2.Hormone <- Disease=="Disease2" & Treatment=="Hormone"
> design <- cbind(design, Healthy.Hormone, Disease1.Hormone, Disease2.Hormone)
```

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

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

To find genes responding to the hormone in Healthy patients:

```
> qlf <- glmQLFTest(fit, coef="Healthy.Hormone")
> topTags(qlf)
```

To find genes responding to the hormone in Disease1 patients:

```
> qlf <- glmQLFTest(fit, coef="Disease1.Hormone")
> topTags(qlf)
```

To find genes responding to the hormone in Disease2 patients:

```
> qlf <- glmQLFTest(fit, coef="Disease2.Hormone")
> topTags(qlf)
```

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

```
> qlf <- glmQLFTest(fit, coef=10:12)
> topTags(qlf)
```

To find genes that respond differently to the hormone in Disease1 vs Healthy patients:

```
> qlf <- glmQLFTest(fit, contrast=c(0,0,0,0,0,0,0,0,0,-1,1,0))
> topTags(qlf)
```

To find genes that respond differently to the hormone in Disease2 vs Healthy patients:

```
> qlf <- glmQLFTest(fit, contrast=c(0,0,0,0,0,0,0,0,0,-1,0,1))
> topTags(qlf)
```

To find genes that respond differently to the hormone in Disease2 vs Disease1 patients:

```
> qlf <- glmQLFTest(fit, contrast=c(0,0,0,0,0,0,0,0,0,0,-1,1))
> topTags(qlf)
```

Chapter 4

Case studies

4.1 RNA-Seq of oral carcinomas vs matched normal tissue

4.1.1 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 [48]. The aim of the analysis is to detect genes differentially expressed between tumor and normal tissue, adjusting for any differences between the patients. This provides an example of the GLM capabilities of edgeR.

RNA was sequenced on an Applied Biosystems SOLiD System 3.0 and reads mapped to the UCSC hg18 reference genome [48]. Read counts, summarised at the level of refSeq transcripts, are available in Table S1 of Tuch et al. [48].

4.1.2 Reading in the data

The read counts for the six individual libraries are stored in one tab-delimited file. To make this file, we downloaded Table S1 from Tuch et al. [48], deleted some unnecessary columns and edited the column headings slightly:

```
> rawdata <- read.delim("TableS1.txt", check.names=FALSE, stringsAsFactors=FALSE)
```

```
> head(rawdata)
```

```
      RefSeqID    Symbol NbrOfExons   8N 8T   33N 33T   51N  51T
1    NM_182502 TMPRSS11B         10 2592  3  7805 321  3372    9
2    NM_003280     TNNC1          6 1684  0  1787   7  4894  559
3    NM_152381     XIRP2         10 9915 15 10396  48 23309 7181
4    NM_022438       MAL          3 2496  2  3585 239  1596    7
5 NM_001100112      MYH2         40 4389  7  7944  16  9262 1818
6    NM_017534      MYH2         40 4402  7  7943  16  9244 1815
```

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

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

4.1.3 Annotation

The study by Tuch et al. [48] was undertaken a few years ago, so not all of the RefSeq IDs provided by match RefSeq IDs currently in use. We retain only those transcripts with IDs in the current NCBI annotation, which is provided by the `org.HS.eg.db` package:

```
> library(org.Hs.eg.db)
> idfound <- y$genes$RefSeqID %in% mappedRkeys(org.Hs.egREFSEQ)
> y <- y[idfound,]
> dim(y)
[1] 15534     6
```

We add Entrez Gene IDs to the annotation:

```
> egREFSEQ <- toTable(org.Hs.egREFSEQ)
> head(egREFSEQ)
  gene_id    accession
1       1    NM_130786
2       1    NP_570602
3       2    NM_000014
4       2 NM_001347423
5       2 NM_001347424
6       2 NM_001347425
> m <- match(y$genes$RefSeqID, egREFSEQ$accession)
> y$genes$EntrezGene <- egREFSEQ$gene_id[m]
```

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

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

```
5 NM_001100112      MYH2         40       4620
6    NM_017534      MYH2         40       4620
```

4.1.4 Filtering and normalization

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

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

[1] 10510
```

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

Recompute the library sizes:

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

Use Entrez Gene IDs as row names:

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

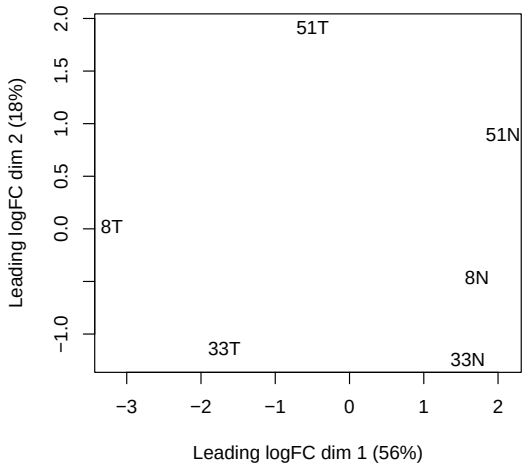
TMM normalization is applied to this dataset to account for compositional difference between the libraries.

```
> y <- normLibSizes(y)
> y$samples

    group lib.size norm.factors
8N      1  7987830        1.146
8T      1  7370197        1.086
33N     1 15752765        0.672
33T     1 14042177        0.973
51N     1 21536577        1.032
51T     1 15191722        1.190
```

4.1.5 Data exploration

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

```
> plotMDS(y)
```



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

4.1.6 Design matrix

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

```
> Patient <- factor(c(8,8,33,33,51,51))
> Tissue <- factor(c("N","T","N","T","N","T"))
> data.frame(Sample=colnames(y),Patient,Tissue)

  Sample Patient Tissue
1     8N       8      N
2     8T       8      T
3    33N      33      N
4    33T      33      T
5    51N      51      N
6    51T      51      T

> design <- model.matrix(~Patient+Tissue)
> rownames(design) <- colnames(y)
> design

    (Intercept) Patient33 Patient51 TissueT
8N            1         0         0       0
8T            1         0         0       1
33N           1         1         0       0
33T           1         1         0       1
51N           1         0         1       0
51T           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 sort of additive model is appropriate for paired designs, or experiments with batch effects.

4.1.7 Dispersion estimation

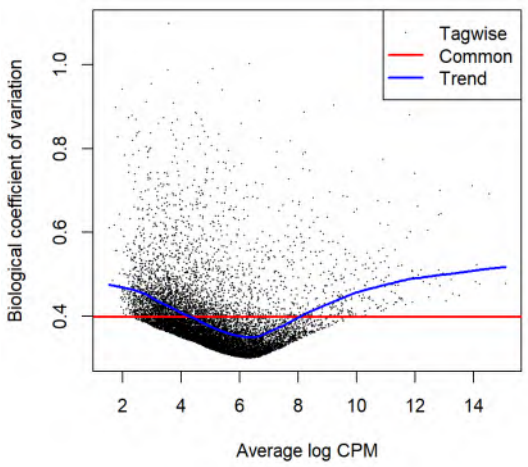
We estimate the NB dispersion for the dataset.

```
> y <- estimateDisp(y, design, robust=TRUE)
> y$common.dispersion

[1] 0.159
```

The square root of the common dispersion gives the coefficient of variation of biological variation. Here the common dispersion is found to be 0.159, so the coefficient of biological variation is around 0.4.

The dispersion estimates can be viewed in a BCV plot:

```
> plotBCV(y)
```



4.1.8 Differential expression

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

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

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

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

Coefficient:  TissueT
          RefSeqID   Symbol NbrOfExons logFC logCPM   LR   PValue      FDR
5737   NM_001039585    PTGFR          4 -5.18   4.74 98.7 2.98e-23 3.13e-19
5744      NM_002820    PTHLH          4  3.97   6.21 92.1 8.12e-22 4.27e-18
3479   NM_001111283     IGF1          5 -3.99   5.72 86.5 1.39e-20 4.86e-17
1288      NM_033641   COL4A6         45  3.66   5.72 77.5 1.32e-18 3.46e-15
10351     NM_007168    ABCA8         38 -3.98   4.94 75.9 2.98e-18 6.27e-15
5837      NM_005609     PYGM         20 -5.48   5.99 75.4 3.93e-18 6.89e-15
487       NM_004320   ATP2A1         23 -4.62   5.96 74.8 5.20e-18 7.81e-15
27179     NM_014440    IL36A          4 -6.17   5.40 72.2 1.95e-17 2.56e-14
196374    NM_173352    KRT78          9 -4.25   7.61 70.8 3.95e-17 4.61e-14
83699     NM_031469 SH3BGRL2          4 -3.93   5.54 67.8 1.84e-16 1.93e-13
```

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

```
> colnames(design)

[1] "(Intercept)" "Patient33"   "Patient51"   "TissueT"
```

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

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

```
> o <- order(lrt$table$PValue)
> cpm(y)[o[1:10],]

           8N      8T    33N     33T    51N      51T
5737    49.70   0.875  27.10   0.878  78.13   2.5435
5744     7.32  95.858  11.80 204.176   6.89 116.3396
3479    50.25   3.124  32.39   1.902 211.65  14.2107
1288    12.12 140.226   6.33  94.443   4.86  56.8427
10351   52.65   3.124  39.48   2.122  79.21   6.0824
5837   152.82   2.750 119.65   1.170  97.70   5.6953
487    107.92   3.124 147.13   3.804 102.83   8.9024
27179   40.09   1.250 172.26   3.292  36.09   0.0553
196374 372.27  20.746 581.55  47.770 145.09   4.5341
83699   96.23   5.124 117.20   5.413  48.20   5.4189
```

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

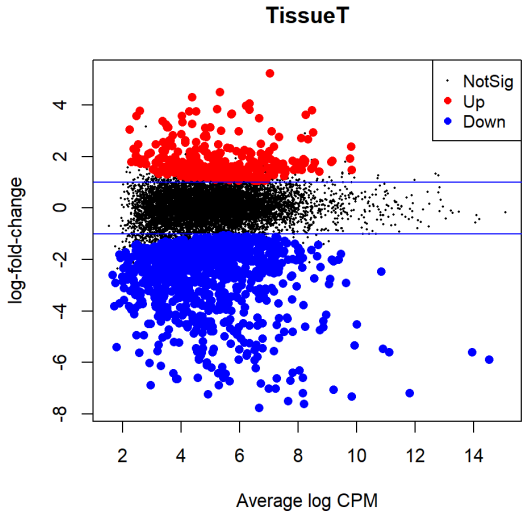
The total number of differentially expressed genes at 5% FDR is given by:

```
> summary(decideTests(lrt))
```

```
       TissueT
Down       938
NotSig    9241
Up         331
```

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

```
> plotMD(lrt)
> abline(h=c(-1, 1), col="blue")
```



The blue lines indicate 2-fold changes.

4.1.9 Gene ontology analysis

We perform a gene ontology analysis focusing on the ontology of biological process (BP). The genes up-regulated in the tumors tend to be associated with cell differentiation, cell migration and tissue morphogenesis:

```
> go <- goana(lrt)
> topGO(go, ont="BP", sort="Up", n=30, truncate=30)
                                    Term Ont    N  Up Down     P.Up   P.Down
GO:0009888             tissue development  BP 1245  82  192 2.66e-11 1.31e-15
GO:0007155                  cell adhesion  BP  928  66  160 1.56e-10 1.95e-17
GO:0022008                   neurogenesis  BP 1024  70  114 2.24e-10 6.42e-03
GO:0007399     nervous system development  BP 1467  87  154 1.34e-09 1.42e-02
GO:0060429         epithelium development  BP  743  54   93 4.66e-09 4.10e-04
GO:0048513       animal organ development  BP 1786  98  265 5.57e-09 1.11e-19
GO:0007275 multicellular organism deve...  BP 2752 133  323 1.18e-08 2.71e-09
GO:0008544          epidermis development  BP  249  27   33 1.86e-08 1.36e-02
GO:0048699          generation of neurons  BP  863  58   99 2.03e-08 4.68e-03
GO:0009653 anatomical structure morpho...  BP 1636  90  240 2.74e-08 5.56e-17
GO:0048731             system development  BP 2321 116  294 2.94e-08 3.35e-12
```

```
GO:0030154           cell differentiation  BP 2501 122  318 4.37e-08 1.33e-13
GO:0048869 cellular developmental proc...  BP 2502 122  318 4.48e-08 1.41e-13
GO:0030155    regulation of cell adhesion  BP  537  41   81 1.12e-07 1.33e-06
GO:0048729           tissue morphogenesis  BP  386  33   48 1.64e-07 1.12e-02
GO:0048856 anatomical structure develo...  BP 3428 152  417 2.10e-07 1.46e-15
GO:0030182         neuron differentiation  BP  815  53   94 2.67e-07 4.98e-03
GO:0016477                cell migration  BP  976  60  151 2.72e-07 2.24e-12
GO:0048870                 cell motility  BP 1047  63  155 2.77e-07 3.80e-11
GO:0043588              skin development  BP  220  23   28 4.22e-07 3.48e-02
GO:0009991 response to extracellular s...  BP  345  30   31 4.29e-07 5.12e-01
GO:0042127 regulation of cell populati...  BP 1015  61  134 4.66e-07 1.18e-06
GO:0008283 cell population proliferati...  BP 1216  69  155 6.11e-07 1.31e-06
GO:0031667    response to nutrient levels  BP  328  28   28 1.53e-06 6.28e-01
GO:0002009 morphogenesis of an epithel...  BP  321  27   34 3.14e-06 1.67e-01
GO:0048598        embryonic morphogenesis  BP  363  29   35 3.76e-06 3.39e-01
GO:0009887     animal organ morphogenesis  BP  577  39   83 4.90e-06 6.65e-06
GO:0030900          forebrain development  BP  238  22   33 5.90e-06 7.12e-03
GO:0007267            cell-cell signaling  BP  857  51  110 6.35e-06 4.17e-05
GO:0032502          developmental process  BP 3701 155  429 6.54e-06 2.21e-12
```

4.1.10 Setup

This analysis was conducted on:

```
> sessionInfo()

R version 4.4.0 (2024-04-24 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 10 x64 (build 19045)

Matrix products: default


locale:
[1] LC_COLLATE=English_Australia.utf8  LC_CTYPE=English_Australia.utf8
[3] LC_MONETARY=English_Australia.utf8 LC_NUMERIC=C
[5] LC_TIME=English_Australia.utf8

time zone: Australia/Sydney
tzcode source: internal

attached base packages:
[1] stats4    stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] org.Hs.eg.db_3.19.1  AnnotationDbi_1.65.2 IRanges_2.37.1
 [4] S4Vectors_0.41.7     Biobase_2.63.1       BiocGenerics_0.49.1
 [7] edgeR_4.1.28         limma_3.59.10        knitr_1.46
[10] BiocStyle_2.31.0
```

```
loaded via a namespace (and not attached):
 [1] bit_4.0.5               jsonlite_1.8.8          compiler_4.4.0
 [4] BiocManager_1.30.22     highr_0.10              crayon_1.5.2
 [7] Rcpp_1.0.12             blob_1.2.4              Biostrings_2.71.6
[10] splines_4.4.0           png_0.1-8               yaml_2.3.8
[13] fastmap_1.1.1           statmod_1.5.0           lattice_0.22-6
[16] R6_2.5.1                XVector_0.43.1          GenomeInfoDb_1.39.14
[19] GenomeInfoDbData_1.2.12 DBI_1.2.2               rlang_1.1.3
[22] KEGGREST_1.43.0         cachem_1.0.8            xfun_0.43
[25] bit64_4.0.5             RSQLite_2.3.6           memoise_2.0.1
[28] cli_3.6.2               zlibbioc_1.49.3         digest_0.6.35
[31] grid_4.4.0              locfit_1.5-9.9          GO.db_3.19.1
[34] vctrs_0.6.5             evaluate_0.23           rmarkdown_2.26
[37] httr_1.4.7              pkgconfig_2.0.3         UCSC.utils_0.99.7
[40] tools_4.4.0             htmltools_0.5.8.1
```

4.2 RNA-Seq of pathogen inoculated arabidopsis with batch effects

4.2.1 Introduction

This case study re-analyses Arabidopsis thaliana RNA-Seq data described by Cumbie et al. [10]. Summarized count data is available as a data object in the CRAN package `NBPSeq` comparing ΔhrcC challenged and mock-inoculated samples [10]. Samples were collected in three batches, and adjustment for batch effects proves to be important. The aim of the analysis therefore is to detect genes differentially expressed in response to ΔhrcC challenge, while correcting for any differences between the batches.

4.2.2 RNA samples

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

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

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

4.2.3 Loading the data

The data is in the NBPSeq package which does not work in R after version 3.5.0. We loaded an earlier version of NBPSeq and saved the data in an RDS file. The RDS file is available here. We then read in the RDS file for our analysis.

```
> library(edgeR)

Loading required package:  limma

> arab <- readRDS("Data/arab.rds")
> head(arab)

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

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

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

We then create a `DGEList` object:

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

4.2.4 Filtering and normalization

There is no purpose in analysing genes that are not expressed in either experimental condition, so genes are first filtered on expression levels.

```
> keep <- filterByExpr(y)
> table(keep)

keep
FALSE  TRUE
12292 13930

> y <- y[keep, , keep.lib.sizes=FALSE]
```

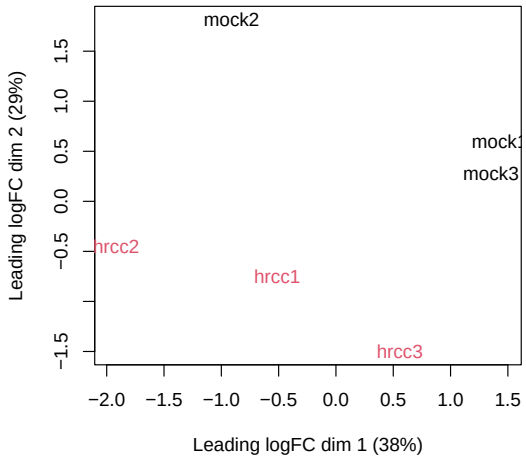
The TMM normalization is applied to account for the compositional biases:

```
> y <- normLibSizes(y)
> y$samples

      group lib.size norm.factors
mock1  mock  1882391        0.977
mock2  mock  1870625        1.023
mock3  mock  3227243        0.914
hrcc1  hrcc  2101449        1.058
hrcc2  hrcc  1243266        1.083
hrcc3  hrcc  3494821        0.955
```

4.2.5 Data exploration

An MDS plot shows the relative similarities of the six samples.

```
> plotMDS(y, col=rep(1:2, each=3))
```



Distances on an MDS plot of a DGEList object correspond to *leading log-fold-change* between each pair of samples. Leading log-fold-change is the root-mean-square average of the largest $\log_2$-fold-changes between each pair of samples. Each pair of samples extracted at each time tend to cluster together, suggesting a batch effect. The hrcc treated samples tend to be below the mock samples for each time, suggesting a treatment effect within each time. The two samples at time 1 are less consistent than at times 2 and 3.

To examine further consistency of the three replicates, we compute predictive log2-fold-changes (logFC) for the treatment separately for the three times.

```
> design <- model.matrix(~Time+Time:Treat)
> logFC <- predFC(y,design,prior.count=1,dispersion=0.05)
```

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

```
> cor(logFC[,4:6])
```

```
                Time1:Treathrcc Time2:Treathrcc Time3:Treathrcc
Time1:Treathrcc           1.000           0.397           0.497
Time2:Treathrcc           0.397           1.000           0.516
Time3:Treathrcc           0.497           0.516           1.000
```

The correlation is highest between times 2 and 3.

4.2.6 Design matrix

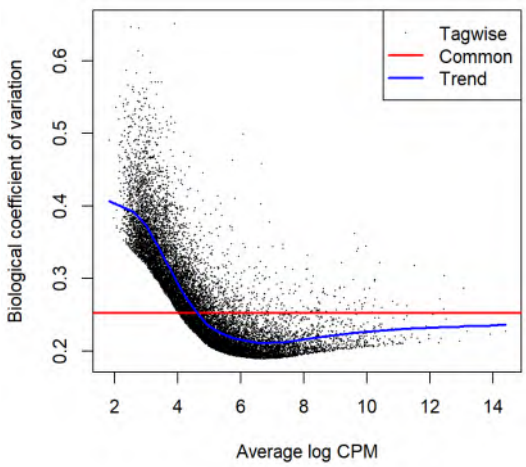
Before we fit GLMs, we need to define our design matrix based on the experimental design. We want to test for differential expressions between ΔhrcC challenged and mock-inoculated samples within batches, i.e. adjusting for differences between batches. In statistical terms, this is an additive linear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

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

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

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

4.2.7 Dispersion estimation

Estimate the genewise dispersion estimates over all genes, allowing for a possible abundance trend. The estimation is also robustified against potential outlier genes.

```
> y <- estimateDisp(y, design, robust=TRUE)
> y$common.dispersion

[1] 0.0638

> plotBCV(y)
```

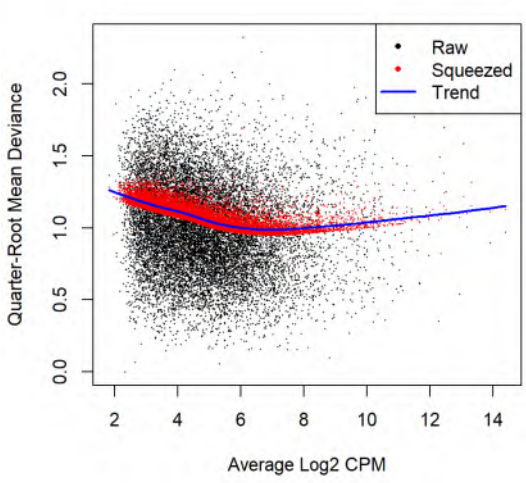


The square root of dispersion is the coefficient of biological variation (BCV). The common BCV is on the high side, considering that this is a designed experiment using genetically identical plants. The trended dispersion shows a decreasing trend with expression level. At low logCPM, the dispersions are very large indeed.

Note that only the trended dispersion is used under the quasi-likelihood (QL) pipeline. The tagwise and common estimates are shown here but will not be used further.

The QL dispersions can be estimated using the `glmQLFit` function, and then be visualized with the `plotQLDisp` function.

```
> fit <- glmQLFit(y, design, robust=TRUE)
> plotQLDisp(fit)
```



4.2.8 Differential expression

Now we test for significant differential expression in each gene using the QL F-test.

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

```
> qlf <- glmQLFTest(fit, coef=2:3)
> topTags(qlf)

Coefficient:  Time2 Time3
          logFC.Time2 logFC.Time3 logCPM     F   PValue      FDR
AT3G33004        4.85      -1.795   5.67 114.2 1.52e-09 1.27e-05
AT5G31702        5.88      -2.611   5.98 122.6 1.82e-09 1.27e-05
AT2G11230        3.53      -1.566   5.64 101.6 3.31e-09 1.54e-05
AT2G07782        3.52      -1.650   5.32  95.3 5.12e-09 1.78e-05
AT2G18193        3.11      -2.425   5.11  86.4 9.43e-09 2.53e-05
AT2G23910        3.64      -0.410   5.17  85.0 1.09e-08 2.53e-05
AT5G54830        3.12      -0.393   6.11  81.8 1.39e-08 2.77e-05
AT2G27770        2.52      -1.592   5.46  76.4 2.19e-08 3.82e-05
AT4G05635        3.21      -2.469   4.80  68.2 4.60e-08 7.12e-05
AT1G05680        2.13      -1.315   6.02  65.5 5.93e-08 8.27e-05

> FDR <- p.adjust(qlf$table$PValue, method="BH")
> sum(FDR < 0.05)

[1] 1707
```

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

```
> qlf <- glmQLFTest(fit)
> topTags(qlf)

Coefficient:  Treathrcc
          logFC logCPM   F   PValue      FDR
AT2G19190  4.48   7.38 295 5.47e-11 7.47e-07
AT2G39530  4.32   6.71 264 1.20e-10 7.47e-07
AT3G46280  4.77   8.10 252 1.61e-10 7.47e-07
AT1G51800  3.95   7.71 231 2.90e-10 8.24e-07
AT2G39380  4.92   5.77 232 2.96e-10 8.24e-07
AT5G48430  6.29   6.74 223 3.75e-10 8.70e-07
AT3G55150  5.75   4.91 191 4.54e-10 9.04e-07
AT1G51850  5.28   5.42 201 6.30e-10 1.10e-06
AT5G64120  3.69   9.70 199 8.08e-10 1.10e-06
AT2G44370  5.40   5.20 191 8.35e-10 1.10e-06
```

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

```
> top <- rownames(topTags(qlf))
> cpm(y)[top,]

            mock1  mock2  mock3  hrcc1  hrcc2  hrcc3
```

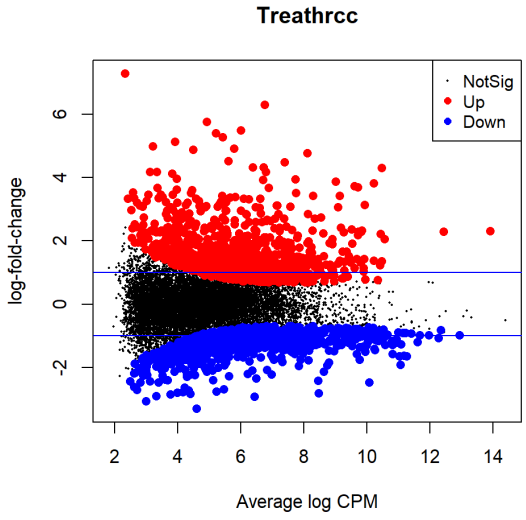
```
AT2G19190  16.853  12.54  13.22  341.7  262.2  344.9
AT2G39530   7.067   9.41  13.22  158.3  197.6  238.8
AT3G46280  19.028  17.77  18.30  385.3  385.5  806.4
AT1G51800  29.357  17.25  30.50  362.8  358.0  455.8
AT2G39380   2.175   3.14   4.75   91.7   86.9  132.8
AT5G48430   4.349   4.70   0.00  189.3  323.9  122.9
AT3G55150   0.544   1.05   1.36   43.2   66.9   63.2
AT1G51850   1.087   1.05   3.73   78.2   57.9  107.0
AT5G64120 135.369 119.68 104.05 1342.5 1692.8 1606.2
AT2G44370   2.175   1.05   1.69   57.1   69.1   84.5
```

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

```
> summary(decideTests(qlf))

       Treathrcc
Down        1019
NotSig     11930
Up           981
```

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

```
> plotMD(qlf)
> abline(h=c(-1,1), col="blue")
```



The blue lines indicate 2-fold up or down.

4.2.9 Setup

This analysis was conducted on:

```
> sessionInfo()
```

```
R version 4.4.0 (2024-04-24 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 10 x64 (build 19045)

Matrix products: default


locale:
[1] LC_COLLATE=English_Australia.utf8  LC_CTYPE=English_Australia.utf8
[3] LC_MONETARY=English_Australia.utf8 LC_NUMERIC=C
[5] LC_TIME=English_Australia.utf8

time zone: Australia/Sydney
tzcode source: internal

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

other attached packages:
[1] edgeR_4.1.28     limma_3.59.10    knitr_1.46       BiocStyle_2.31.0

loaded via a namespace (and not attached):
 [1] digest_0.6.35       fastmap_1.1.1       xfun_0.43
 [4] lattice_0.22-6      splines_4.4.0       htmltools_0.5.8.1
 [7] rmarkdown_2.26      cli_3.6.2           grid_4.4.0
[10] statmod_1.5.0       compiler_4.4.0      highr_0.10
[13] tools_4.4.0         evaluate_0.23       Rcpp_1.0.12
[16] yaml_2.3.8          locfit_1.5-9.9      BiocManager_1.30.22
[19] rlang_1.1.3
```

4.3 Profiles of Yoruba HapMap individuals

4.3.1 Background

RNA-Seq profiles were made of cell lines derived from lymphoblastoid cells from 69 different Yoruba individuals from Ibadan, Nigeria [36] [37]. The profiles were generated as part of the International HapMap project [21]. RNA from each individual was sequenced on at least two lanes of an Illumina Genome Analyser 2, and mapped reads to the human genome using MAQ v0.6.8.

The study group here is essentially an opportunity sample and the individuals are likely to be genetically diverse. In this analysis we look at genes that are differentially expressed between males and female.

4.3.2 Loading the data

Read counts summarized by Ensembl gene identifiers are available in the tweeDEseqCountData package:

```
> library(tweeDEseqCountData)
> data(pickrell1)
> Counts <- exprs(pickrell1.eset)
> dim(Counts)

[1] 38415    69

> Counts[1:5,1:5]

                NA18486 NA18498 NA18499 NA18501 NA18502
ENSG00000127720       6      32      14      35      14
ENSG00000242018      20      21      24      22      16
ENSG00000224440       0       0       0       0       0
ENSG00000214453       0       0       0       0       0
ENSG00000237787       0       0       1       0       0
```

In this analysis we will compare female with male individuals.

```
> Gender <- pickrell1.eset$gender
> table(Gender)

Gender
female   male
    40     29

> rm(pickrell1.eset)
```

Annotation for each Ensemble gene is also available from the tweeDEseqCountData package:

```
> data(annotEnsembl63)
> annot <- annotEnsembl63[,c("Symbol","Chr")]
> annot[1:5,]

                Symbol Chr
ENSG00000252775     U7   5
ENSG00000207459     U6   5
ENSG00000252899     U7   5
ENSG00000201298     U6   5
ENSG00000222266     U6   5

> rm(annotEnsembl63)
```

Form a DGEList object combining the counts and associated annotation:

```
> library(edgeR)
> y <- DGEList(counts=Counts, genes=annot[rownames(Counts),])
```

4.3.3 Filtering and normalization

Keep genes that are expressed in a worthwhilte number of samples:

```
> isexpr <- filterByExpr(y, group=Gender)
> table(isexpr)

isexpr
```

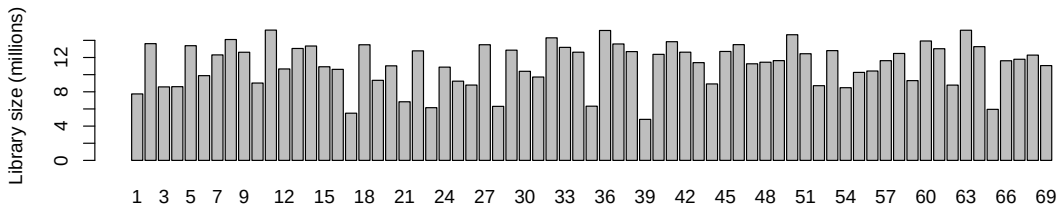
```
FALSE  TRUE
20226 18189
```

Keep only genes with defined annotation, and recompute library sizes:

```
> hasannot <- rowSums(is.na(y$genes))==0
> y <- y[isexpr & hasannot, , keep.lib.sizes=FALSE]
> dim(y)

[1] 17517    69
```

The library sizes vary from about 5 million to over 15 million:

```
> barplot(y$samples$lib.size*1e-6, names=1:69, ylab="Library size (millions)")
```



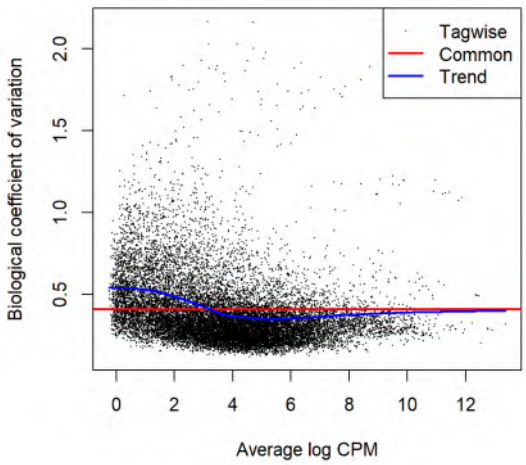
Apply TMM normalization to account for the composition biases:

```
> y <- normLibSizes(y)
> head(y$samples)

        group lib.size norm.factors
NA18486     1  7750614        0.929
NA18498     1 13614927        1.096
NA18499     1  8570996        0.958
NA18501     1  8596932        1.194
NA18502     1 13377004        0.942
NA18504     1  9883172        0.983
```

4.3.4 Dispersion estimation

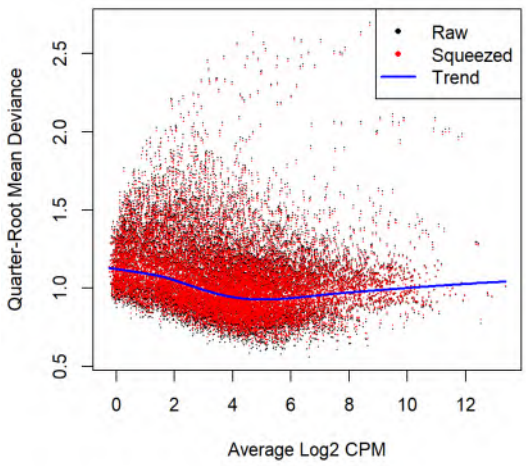
We are interested in the differences between male and female. Hence, we create a design matrix using the gender factor. We estimate the NB dispersion using `estimateDisp`. The estimation is robustified against potential outlier genes. Note that this step is now optional as all the NB dispersion estimates will not be used further under the latest quasi-likelihood (QL) pipeline.

```
> design <- model.matrix(~Gender)
> y <- estimateDisp(y, design, robust=TRUE)
> plotBCV(y)
```



We then estimate the QL dispersions using `glmQLFit`. The large number of cases and the high variability means that the QL dispersions are not squeezed very heavily from the raw values:

```
> fit <- glmQLFit(y, design, robust=TRUE)
> plotQLDisp(fit)
```



4.3.5 Differential expression

Now find genes differentially expressed between male and females. Positive log-fold-changes mean higher expression in males. The highly ranked genes are mostly on the X or Y chromosomes. Top ranked is the famous XIST gene, which is known to be expressed only in females.

```
> qlf <- glmQLFTest(fit)
> topTags(qlf, n=15)

Coefficient:  Gendermale
                     Symbol Chr logFC logCPM    F   PValue      FDR
ENSG00000229807          XIST   X -9.49  7.249 1441 2.58e-51 4.53e-47
ENSG00000233864        TTTY15   Y  4.85  1.254  742 1.17e-49 1.03e-45
ENSG00000131002      CYorf15B   Y  5.63  2.056  770 1.98e-49 1.16e-45
ENSG00000099749      CYorf15A   Y  4.29  1.757 1082 9.03e-48 3.96e-44
ENSG00000165246        NLGN4Y   Y  5.11  1.675  422 1.64e-37 5.74e-34
ENSG00000157828        RPS4Y2   Y  3.18  4.208  611 7.38e-37 2.15e-33
ENSG00000198692        EIF1AY   Y  2.36  3.247  401 4.22e-31 1.06e-27
ENSG00000183878           UTY   Y  1.86  3.137  273 3.45e-26 7.56e-23
ENSG00000243209    AC010889.1   Y  2.65  0.797  250 1.94e-25 3.77e-22
ENSG00000213318  RP11-331F4.1  16  3.67  3.688  242 1.56e-24 2.73e-21
ENSG00000129824        RPS4Y1   Y  2.53  5.401  236 2.07e-24 3.30e-21
ENSG00000012817         KDM5D   Y  1.47  4.949  232 3.24e-24 4.73e-21
ENSG00000146938        NLGN4X   X  3.95  1.047  194 6.62e-24 8.92e-21
ENSG00000067048         DDX3Y   Y  1.62  5.621  188 9.43e-22 1.18e-18
ENSG00000232928 RP13-204A15.4   X  1.44  3.558  117 1.01e-16 1.18e-13

> summary(decideTests(qlf))

       Gendermale
Down           49
NotSig      17447
Up             21
```

4.3.6 Gene set testing

The tweeDEseqCountData package includes a list of genes belonging to the male-specific region of chromosome Y, and a list of genes located in the X chromosome that have been reported to escape X-inactivation. We expect genes in the first list to be up-regulated in males, whereas genes in the second list should be up-regulated in females.

```
> data(genderGenes)
> Ymale <- rownames(y) %in% msYgenes
> Xescape <- rownames(y) %in% XiEgenes
```

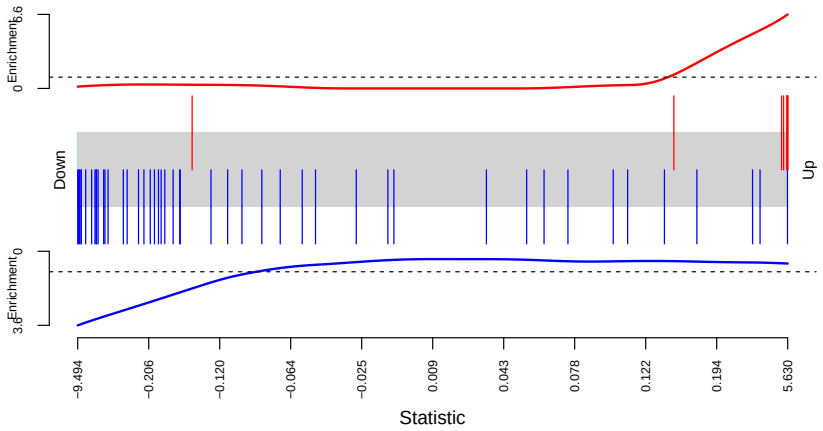
Roast gene set tests by `fry()` confirm that the male-specific genes are significantly up as a group in our comparison of males with females, whereas the X genes are significantly down as a group [49].

```
> index <- list(Y=Ymale, X=Xescape)
> fry(y, index=index, design=design)

  NGenes Direction   PValue      FDR PValue.Mixed FDR.Mixed
Y     12        Up 1.00e-45 2.01e-45     6.70e-11  6.70e-11
X     47      Down 6.93e-17 6.93e-17     1.26e-68  2.53e-68
```

A barcode plot can be produced to visualize the results. Genes are ranked from left to right by increasing log-fold-change in the background of the barcode plot. Genes in the set of `msYgenes` are represented by red bars whereas genes in the set of `XiEgenes` are represented by

blue bars. The line above the barcode shows the relative local enrichment of the vertical bars in each part of the plot. This particular plot suggests that the male-specific genes tend to have large positive log-fold-changes, whereas the X genes tend to have large negative log-fold-changes.

```
> barcodeplot(qlf$table$logFC, index[[1]], index[[2]])
```

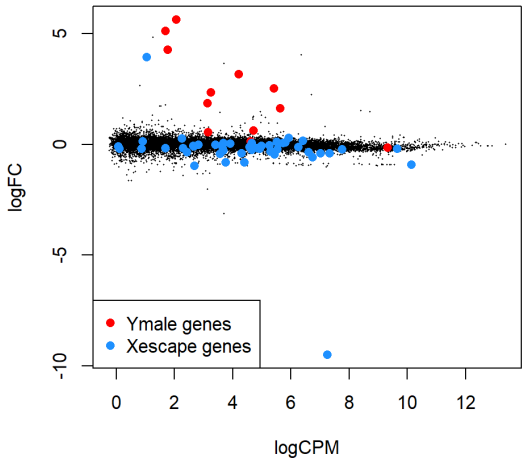


The results from competitive camera gene sets tests are even more convincing [50]. The positive intergene correlations here show that the genes in each set tend to be biologically correlated:

```
> camera(y, index, design)

  NGenes Direction    PValue       FDR
Y     12        Up 1.32e-295 2.65e-295
X     47      Down  7.38e-25  7.38e-25
```

See where the X and Y genes fall on the MA plot:

```
> with(qlf$table, plot(logCPM,logFC,pch=16,cex=0.2))
> with(qlf$table, points(logCPM[Ymale],logFC[Ymale],pch=16,col="red"))
> with(qlf$table, points(logCPM[Xescape],logFC[Xescape],pch=16,col="dodgerblue"))
> legend("bottomleft",legend=c("Ymale genes","Xescape genes"),
+     pch=16,col=c("red","dodgerblue"))
```



4.3.7 Setup

This analysis was conducted on:

```
> sessionInfo()

R version 4.4.0 (2024-04-24 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 10 x64 (build 19045)

Matrix products: default


locale:
[1] LC_COLLATE=English_Australia.utf8  LC_CTYPE=English_Australia.utf8
[3] LC_MONETARY=English_Australia.utf8 LC_NUMERIC=C
[5] LC_TIME=English_Australia.utf8

time zone: Australia/Sydney
tzcode source: internal

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

other attached packages:
[1] edgeR_4.1.28              limma_3.59.10
[3] tweeDEseqCountData_1.41.0 Biobase_2.63.1
[5] BiocGenerics_0.49.1       knitr_1.46
[7] BiocStyle_2.31.0

loaded via a namespace (and not attached):
 [1] digest_0.6.35       fastmap_1.1.1       xfun_0.43
 [4] lattice_0.22-6      splines_4.4.0       htmltools_0.5.8.1
```

```
 [7] rmarkdown_2.26      cli_3.6.2           grid_4.4.0
[10] statmod_1.5.0       compiler_4.4.0      highr_0.10
[13] tools_4.4.0         evaluate_0.23       Rcpp_1.0.12
[16] yaml_2.3.8          locfit_1.5-9.9      BiocManager_1.30.22
[19] rlang_1.1.3
```

4.4 RNA-Seq profiles of mouse mammary gland

4.4.1 Introduction

The RNA-Seq data of this case study is described in Fu *et al.* [17]. The sequence and count data are publicly available from the Gene Expression Omnibus (GEO) at the series accession number GSE60450. This study examines the expression profiles of basal stem-cell enriched cells (B) and committed luminal cells (L) in the mammary gland of virgin, pregnant and lactating mice. Six groups are present, with one for each combination of cell type and mouse status. Each group contains two biological replicates. This is summarized in the table below, where the basal and luminal cell types are abbreviated with `B` and `L` respectively.

```
> targets <- read.delim("targets.txt", header=TRUE)
```

```
> targets
           FileName GEOAccession CellType   Status
1  SRR1552450.fastq   GSM1480297        B   virgin
2  SRR1552451.fastq   GSM1480298        B   virgin
3  SRR1552452.fastq   GSM1480299        B pregnant
4  SRR1552453.fastq   GSM1480300        B pregnant
5  SRR1552454.fastq   GSM1480301        B  lactate
6  SRR1552455.fastq   GSM1480302        B  lactate
7  SRR1552444.fastq   GSM1480291        L   virgin
8  SRR1552445.fastq   GSM1480292        L   virgin
9  SRR1552446.fastq   GSM1480293        L pregnant
10 SRR1552447.fastq   GSM1480294        L pregnant
11 SRR1552448.fastq   GSM1480295        L  lactate
12 SRR1552449.fastq   GSM1480296        L  lactate
```

The name of the file containing the read sequences for each library is also shown. Each file is downloaded from the Sequence Read Archive and has an accession number starting with SRR, e.g., SRR1552450 for the first library in `targets`.

4.4.2 Read alignment and processing

Prior to read alignment, these files are converted into the FASTQ format using the `fastq-dump` utility from the SRA Toolkit. See https://www.ncbi.nlm.nih.gov/books/NBK158900 for how to download and use the SRA Toolkit.

Before the differential expression analysis can proceed, these reads must be aligned to the mouse genome and counted into annotated genes. This can be achieved with functions in the Rsubread package [24]. We assume that an index of the mouse genome is already available -

if not, this can be constructed from a FASTA file of the genome sequence with the `buildindex` command. In this example, we assume that the prefix for the index files is `mm10`. The reads in each FASTQ file are then aligned to the mouse genome, as shown below.

```
> library(Rsubread)
> output.files <- sub(".fastq", ".bam", targets$FileName)
> align("mm10", readfile1=targets$FileName, phredOffset=33,
+     input_format="FASTQ", output_file=output.files)
```

This produces a set of BAM files, where each file contains the read alignments for each library. The mapped reads can be counted into mouse genes by using the `featureCounts` function. It uses the exon intervals defined in the NCBI annotation of the mm10 genome.

```
> fc <- featureCounts(output.files, annot.inbuilt="mm10")
> colnames(fc$counts) <- 1:12
```

```
> head(fc$counts)

            1   2   3   4   5   6   7   8   9  10  11  12
497097    438 300  65 237 354 287   0   0   0   0   0   0
100503874   1   0   1   1   0   4   0   0   0   0   0   0
100038431   0   0   0   0   0   0   0   0   0   0   0   0
19888       1   1   0   0   0   0  10   3  10   2   0   0
20671     106 182  82 105  43  82  16  25  18   8   3  10
27395     309 234 337 300 290 270 560 464 489 328 307 342
```

The row names of the matrix represent the Entrez gene identifiers for each gene. In the output from `featureCounts`, the column names of `fc$counts` are the output file names from `align`. Here, we simplify them for brevity.

4.4.3 Count loading and annotation

We create a `DGEList` object as follows

```
> group <- factor(paste0(targets$CellType, ".", targets$Status))
> y <- DGEList(fc$counts, group=group)
> colnames(y) <- targets$GEO
```

Human-readable gene symbols can also be added to complement the Entrez identifiers for each gene, using the annotation in the `org.Mm.eg.db` package.

```
> require(org.Mm.eg.db)
> Symbol <- mapIds(org.Mm.eg.db, keys=rownames(y), keytype="ENTREZID",
+     column="SYMBOL")
> y$genes <- data.frame(Symbol=Symbol)
```

4.4.4 Filtering and normalization

Here, a gene is only retained if it is expressed at a minimum level:

```
> keep <- filterByExpr(y)
> summary(keep)
```

```
   Mode   FALSE    TRUE
logical   11210   15969
```

```
> y <- y[keep, , keep.lib.sizes=FALSE]
```

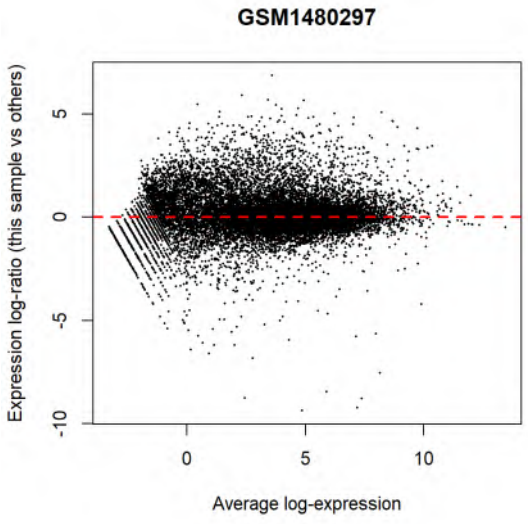
TMM normalization is performed to eliminate composition biases between libraries.

```
> y <- normLibSizes(y)
> y$samples
```

```
               group lib.size norm.factors
GSM1480297   B.virgin 23219195        1.238
GSM1480298   B.virgin 21769326        1.214
GSM1480299 B.pregnant 24092719        1.125
GSM1480300 B.pregnant 22657703        1.071
GSM1480301  B.lactate 21522881        1.036
GSM1480302  B.lactate 20009184        1.087
GSM1480291   L.virgin 20385437        1.368
GSM1480292   L.virgin 21699830        1.365
GSM1480293 L.pregnant 22236469        1.004
GSM1480294 L.pregnant 21983364        0.923
GSM1480295  L.lactate 24720123        0.529
GSM1480296  L.lactate 24653390        0.535
```

The performance of the TMM normalization procedure can be examined using mean-difference (MD) plots. This visualizes the library size-adjusted log-fold change between two libraries (the difference) against the average log-expression across those libraries (the mean). The following MD plot is generated by comparing sample 1 against an artificial library constructed from the average of all other samples.

```
> plotMD(cpm(y, log=TRUE), column=1)
> abline(h=0, col="red", lty=2, lwd=2)
```

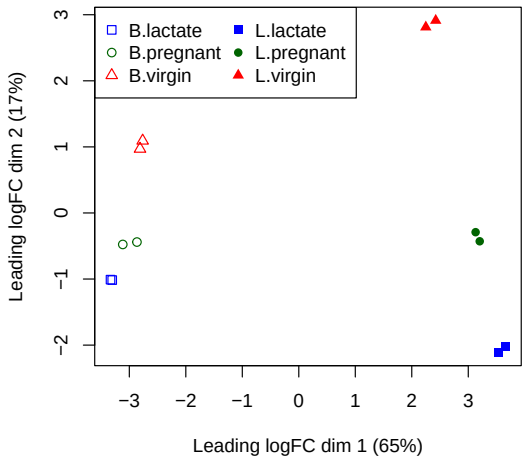


Ideally, the bulk of genes should be centred at a log-fold change of zero. This indicates that any composition bias between libraries has been successfully removed. This quality check should be repeated by constructing a MD plot for each sample.

4.4.5 Data exploration

The data can be explored by generating multi-dimensional scaling (MDS) plots. This visualizes the differences between the expression profiles of different samples in two dimensions.

```
> points <- c(0,1,2,15,16,17)
> colors <- rep(c("blue", "darkgreen", "red"), 2)
> plotMDS(y, col=colors[group], pch=points[group])
> legend("topleft", legend=levels(group), pch=points, col=colors, ncol=2)
```



Replicate samples from the same group cluster together in the plot, while samples from different groups form separate clusters. This indicates that the differences between groups are larger than those within groups, i.e., differential expression is greater than the variance and can be detected. The distance between basal samples on the left and luminal cells on the right is about 6 units, corresponding to a leading fold change of about 64-fold ($2^6 = 64$) between basal and luminal. The expression differences between virgin, pregnant and lactating are greater for luminal cells than for basal.

4.4.6 Design matrix

The experimental design for this study can be parametrized with a one-way layout, whereby one coefficient is assigned to each group. The design matrix contains the predictors for each sample and and is constructed using the code below.

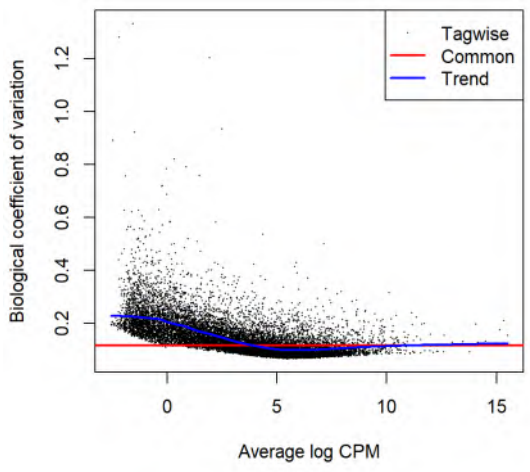
```
> design <- model.matrix(~ 0 + group)
> colnames(design) <- levels(group)
> design

   B.lactate B.pregnant B.virgin L.lactate L.pregnant L.virgin
```

```
1          0          0          1          0          0          0
2          0          0          1          0          0          0
3          0          1          0          0          0          0
4          0          1          0          0          0          0
5          1          0          0          0          0          0
6          1          0          0          0          0          0
7          0          0          0          0          0          1
8          0          0          0          0          0          1
9          0          0          0          0          1          0
10         0          0          0          0          1          0
11         0          0          0          1          0          0
12         0          0          0          1          0          0
attr(,"assign")
[1] 1 1 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$group
[1] "contr.treatment"
```

4.4.7 Dispersion estimation

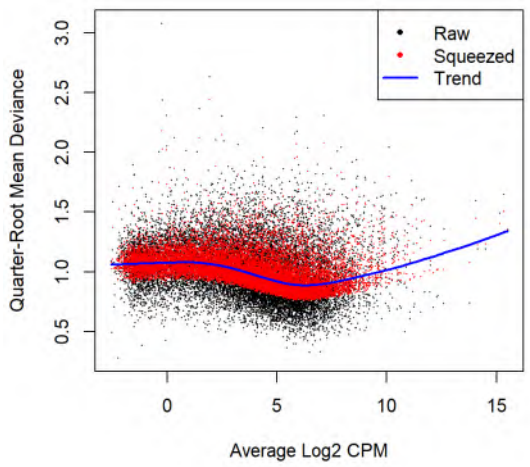
The NB dispersions are estimated using the `estimateDisp` function. This returns the `DGEList` object with additional entries for the estimated NB dispersions for all gene. Note that this step is now optional as all the NB dispersion estimates will not be used further under the latest quasi-likelihood (QL) pipeline. These estimates can be visualized with `plotBCV`, which shows the root-estimate, i.e., the biological coefficient of variation for each gene.

```
> y <- estimateDisp(y, design, robust=TRUE)
> y$common.dispersion

[1] 0.0134

> plotBCV(y)
```



Under the latest QL pipeline, technical and biological variation can be estimated simultaneously in the `glmQLFit` function. This returns a `DGEGLM` object containing the estimated values of the GLM coefficients for each gene, an overall estimate of the squared biological coefficient of variation, as well as the fitted mean-QL dispersion trend, the squeezed QL estimates and the prior degrees of freedom (df). These can be visualized with the `plotQLDisp` function.

```
> fit <- glmQLFit(y, design, robust=TRUE)
> fit$dispersion

[1] 0.0125

> head(fit$coefficients)

       B.lactate B.pregnant B.virgin L.lactate L.pregnant L.virgin
497097    -11.14     -12.02   -11.23     -19.0     -19.03    -19.0
20671     -12.77     -12.52   -12.15     -14.5     -14.30    -14.1
27395     -11.27     -11.30   -11.53     -10.6     -10.87    -10.9
18777     -10.15     -10.21   -10.77     -10.1     -10.39    -10.4
21399      -9.89      -9.74    -9.79     -10.2      -9.97    -10.0
58175     -16.16     -14.85   -15.99     -13.3     -12.29    -12.1

> plotQLDisp(fit)
```



Setting `robust=TRUE` in `glmQLFit` is strongly recommended [35]. Setting `robust=TRUE` in `estimateDisp` has no effect on the downstream analysis, but is nevertheless very useful as it identifies genes that are outliers from the mean-NB dispersion trend.

4.4.8 Differential expression

We test for significant differential expression in each gene, using the QL F-test. The contrast of interest can be specified using the `makeContrasts` function. Here, genes are tested for DE between the basal pregnant and lactating groups. This is done by defining the null hypothesis as `B.pregnant - B.lactate = 0`.

```
> con <- makeContrasts(B.pregnant - B.lactate, levels=design)
> qlf <- glmQLFTest(fit, contrast=con)
```

The top set of most significant genes can be examined with `topTags`. Here, a positive log-fold change represents genes that are up in `B.pregnant` over `B.lactate`. Multiplicity correction is performed by applying the Benjamini-Hochberg method on the p-values, to control the false discovery rate (FDR).

```
> topTags(qlf)

Coefficient:  -1*B.lactate 1*B.pregnant
        Symbol logFC logCPM   F   PValue      FDR
211577  Mrgprf -5.15   2.74 478 4.02e-13 6.42e-09
140474    Muc4  7.17   6.05 336 2.85e-11 1.70e-07
12992  Csn1s2b -6.09  10.18 429 3.20e-11 1.70e-07
24117     Wif1  1.82   6.76 292 7.47e-11 2.98e-07
226101    Myof -2.32   6.44 319 1.99e-10 5.44e-07
381290  Atp2b4 -2.14   6.14 318 2.05e-10 5.44e-07
21953    Tnni2 -5.76   3.86 355 3.31e-10 7.55e-07
231830 Micall2  2.25   5.18 279 4.58e-10 9.14e-07
14585    Gfra1  2.26   2.73 197 6.55e-10 1.16e-06
78896    Ecrg4  2.81   6.68 230 7.44e-10 1.19e-06
```

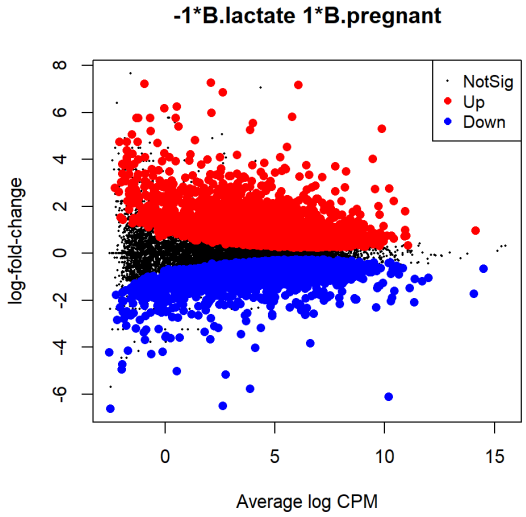
The top gene Csn1s2b has a large negative log2-fold-change, showing that it is far more highly expressed in lactating than pregnant mice. This gene is known to be a major source of protein in milk.

The total number of DE genes in each direction at a FDR of 5% can be examined with `decideTests`. There are in fact nearly 4500 DE genes an FDR cut-off of 5% in this comparison:

```
> summary(decideTests(qlf))

       -1*B.lactate 1*B.pregnant
Down                       2618
NotSig                    10543
Up                         2808
```

The differential expression test results can be visualized using an MD plot. The log-fold change for each gene is plotted against the average abundance, i.e., `logCPM` in the result table above. Significantly DE genes at a FDR of 5% are highlighted.

```
> plotMD(qlf)
```



We use `glmTreat` to narrow down the list of DE genes and focus on genes that are more biologically meaningful. We test whether the differential expression is significantly above a log2-fold-change of $\log_2 1.2$, i.e., a fold-change of 1.2.

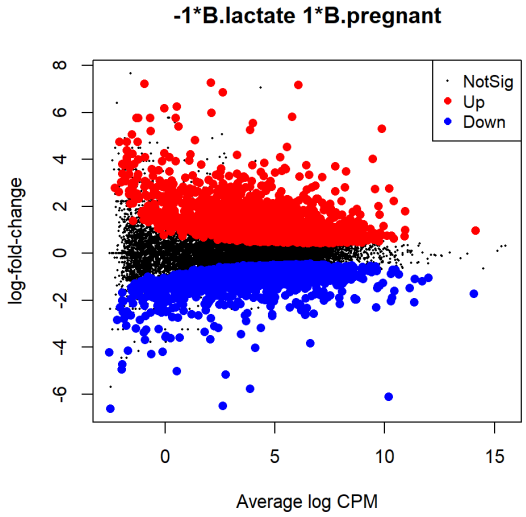
```
> tr <- glmTreat(fit, contrast=con, lfc=log2(1.2))
> topTags(tr)

Coefficient:  -1*B.lactate 1*B.pregnant
        Symbol logFC unshrunk.logFC logCPM   PValue      FDR
211577  Mrgprf -5.15          -5.16   2.74 4.56e-13 7.28e-09
140474    Muc4  7.17           7.34   6.05 3.22e-11 1.81e-07
12992  Csn1s2b -6.09          -6.09  10.18 3.41e-11 1.81e-07
24117     Wif1  1.82           1.82   6.76 1.48e-10 5.90e-07
226101    Myof -2.32          -2.32   6.44 2.85e-10 7.97e-07
381290  Atp2b4 -2.14          -2.15   6.14 3.11e-10 7.97e-07
21953    Tnni2 -5.76          -5.76   3.86 3.49e-10 7.97e-07
231830 Micall2  2.25           2.25   5.18 6.72e-10 1.34e-06
78896    Ecrg4  2.81           2.81   6.68 9.54e-10 1.55e-06
14585    Gfra1  2.26           2.27   2.73 1.09e-09 1.55e-06
```

Around 3000 genes are detected as DE with fold-change significantly above 1.2 at an FDR cut-off of 5%.

```
> summary(decideTests(tr))

       -1*B.lactate 1*B.pregnant
Down                        1541
NotSig                     12580
Up                          1848
```

The test results are visualized in the following smear plot. Genes that are significantly DE above a fold-change of 1.2 at an FDR of 5% are highlighted in red.

```
> plotMD(tr)
```



4.4.9 ANOVA-like testing

The differential expression analysis of two-group comparison can be easily extended to comparisons between three or more groups. This is done by creating a matrix of contrasts, where which each column represents a contrast between two groups of interest. In this manner, users can perform a one-way analysis of variance (ANOVA) for each gene.

As an example, suppose we want to compare the three groups in the luminal population, i.e., virgin, pregnant and lactating. An appropriate contrast matrix can be created as shown below, to make pairwise comparisons between all three groups.

```
> con <- makeContrasts(
+      L.PvsL = L.pregnant - L.lactate,
+      L.VvsL = L.virgin - L.lactate,
+      L.VvsP = L.virgin - L.pregnant, levels=design)
```

The QL F-test is then applied to identify genes that are DE among the three groups. This combines the three pairwise comparisons into a single F-statistic and p-value. The top set of significant genes can be displayed with `topTags`.

```
> anov <- glmQLFTest(fit, contrast=con)
> topTags(anov)

Coefficient:  LR test on 2 degrees of freedom
       Symbol logFC.L.PvsL logFC.L.VvsL logFC.L.VvsP logCPM    F   PValue
19242     Ptn        -1.54         7.26         8.80   7.97 2429 3.09e-17
13645     Egf        -5.36        -7.22        -1.86   3.67 1183 3.21e-15
12992 Csn1s2b        -8.55       -11.36        -2.81  10.18 1148 3.68e-15
52150   Kcnk6        -2.42        -7.00        -4.58   5.91 1003 8.63e-15
83492   Gsdmc        -4.84       -11.35        -6.51   3.16  463 8.85e-15
15439      Hp         1.08         5.42         4.34   4.93 1004 9.24e-15
14183   Fgfr2        -1.15         3.95         5.10   7.38  959 1.15e-14
11941  Atp2b2        -7.37       -10.56        -3.19   6.60 1138 1.45e-14
17068    Ly6d         3.42         9.24         5.82   4.68  884 1.60e-14
```

```
20856    Stc2        -1.81         3.19         5.01   6.10  901 1.71e-14
           FDR
19242 4.93e-13
13645 1.96e-11
12992 1.96e-11
52150 2.46e-11
83492 2.46e-11
15439 2.46e-11
14183 2.59e-11
11941 2.59e-11
17068 2.59e-11
20856 2.59e-11
```

Note that the three contrasts of pairwise comparisons are linearly dependent. Constructing the contrast matrix with any two of the contrasts would be sufficient to specify an ANOVA test. For instance, the contrast matrix shown below produces the same test results but with a different column of log-fold changes.

```
> con <- makeContrasts(
+      L.PvsL = L.pregnant - L.lactate,
+      L.VvsP = L.virgin - L.pregnant, levels=design)
```

4.4.10 Gene ontology analysis

Further analyses are required to interpret the differential expression results in a biological context. One common downstream procedure is a gene ontology (GO) enrichment analysis.

Suppose we want to identify GO terms that are over-represented in the basal lactating group compared to the basal pregnancy group. This can be achieved by applying the `goana` function to the differential expression results of that comparison. The top set of most enriched GO terms can be viewed with the `topGO` function.

```
> con <- makeContrasts(B.lactate - B.pregnant, levels=design)
> qlf <- glmQLFTest(fit, contrast=con)
> go <- goana(qlf, species = "Mm")
> topGO(go, n=30, truncate=30)

                                    Term Ont    N  Up Down  P.Up   P.Down
GO:0022613 ribonucleoprotein complex b...  BP  428  25  211 1.000 2.07e-52
GO:0042254            ribosome biogenesis  BP  313  10  172 1.000 2.41e-51
GO:1990904      ribonucleoprotein complex  CC  694  45  277 1.000 9.84e-46
GO:0022626             cytosolic ribosome  CC  125   2   88 1.000 1.04e-38
GO:0006364                rRNA processing  BP  213   5  121 1.000 2.59e-38
GO:0016072         rRNA metabolic process  BP  248  11  132 1.000 1.46e-37
GO:0030684                    preribosome  CC  110   1   79 1.000 7.06e-36
GO:0002181        cytoplasmic translation  BP  142   6   89 1.000 4.70e-33
GO:0003723                    RNA binding  MF 1029 118  324 1.000 2.04e-29
GO:0003735 structural constituent of r...  MF  163   1   92 1.000 4.14e-29
GO:0044391              ribosomal subunit  CC  196   1  102 1.000 3.25e-28
GO:0022625 cytosolic large ribosomal s...  CC   63   0   51 1.000 6.74e-28
GO:0005840                       ribosome  CC  241   7  115 1.000 2.68e-27
GO:0042274 ribosomal small subunit bio...  BP  102   1   66 1.000 5.12e-26
```

```
GO:0006412                   translation  BP  604  62  212 1.000 5.84e-26
GO:0043043 peptide biosynthetic proces...  BP  626  66  217 1.000 9.82e-26
GO:0006396                RNA processing  BP  871  76  276 1.000 1.63e-25
GO:0034660        ncRNA metabolic process  BP  581  55  205 1.000 1.87e-25
GO:0032991     protein-containing complex  CC 5023 782 1116 0.993 8.84e-25
GO:0034470              ncRNA processing  BP  392  20  154 1.000 9.57e-25
GO:0005730                    nucleolus  CC  853 118  268 0.990 3.67e-24
GO:0032040       small-subunit processome  CC   73   1   52 1.000 8.87e-24
GO:0140236      translation at presynapse  BP   47   0   40 1.000 9.27e-24
GO:0006518      peptide metabolic process  BP  770  85  246 1.000 2.54e-23
GO:0140242     translation at postsynapse  BP   48   0   40 1.000 4.62e-23
GO:0140241         translation at synapse  BP   48   0   40 1.000 4.62e-23
GO:0043604     amide biosynthetic process  BP  727  82  234 1.000 1.09e-22
GO:1901566 organonitrogen compound bio...  BP 1471 171  396 1.000 5.60e-21
GO:0070013 intracellular organelle lum...  CC 4090 636  917 0.986 2.22e-20
GO:0031974        membrane-enclosed lumen  CC 4090 636  917 0.986 2.22e-20
```

The row names of the output are the universal identifiers of the GO terms, with one term per row. The `Term` column gives the names of the GO terms. These terms cover three domains - biological process (BP), cellular component (CC) and molecular function (MF), as shown in the `Ont` column. The `N` column represents the total number of genes that are annotated with each GO term. The `Up` and `Down` columns represent the number of genes with the GO term that are significantly up- and down-regulated in this differential expression comparison, respectively. The `P.Up` and `P.Down` columns contain the p-values for over-representation of the GO term across the set of up- and down-regulated genes, respectively. The output table is sorted by the minimum of `P.Up` and `P.Down` by default.

The `goana` function uses the NCBI RefSeq annotation. Therefore, the Entrez Gene identifier (ID) should be supplied for each gene as the row names of `qlf`.

4.4.11 Gene set testing

Another downstream step uses the rotation gene set test (ROAST) [49]. Given a set of genes, we can test whether the majority of the genes in the set are DE across the contrast of interest. It is useful when the specified set contains all genes involved in some pathway or process.

In our case study, suppose we are interested in two GO terms related to cytokinesis. Each term will be used to define a set containing all genes that are annotated with that term. The names of these terms can be viewed as shown below.

```
> library(GO.db)
> cyt.go <- c("GO:0032465", "GO:0000281")
> term <- select(GO.db, keys=cyt.go, columns="TERM")
> term

        GOID                        TERM
1 GO:0032465 regulation of cytokinesis
2 GO:0000281       mitotic cytokinesis
```

We construct a list of two components, each of which is a vector of Entrez Gene IDs for all genes annotated with one of the GO terms. We then convert the Gene IDs into row indices of the `fit` object using the function `ids2indices`.

```
> Rkeys(org.Mm.egGO2ALLEGS) <- cyt.go
> ind <- ids2indices(as.list(org.Mm.egGO2ALLEGS), row.names(fit))
```

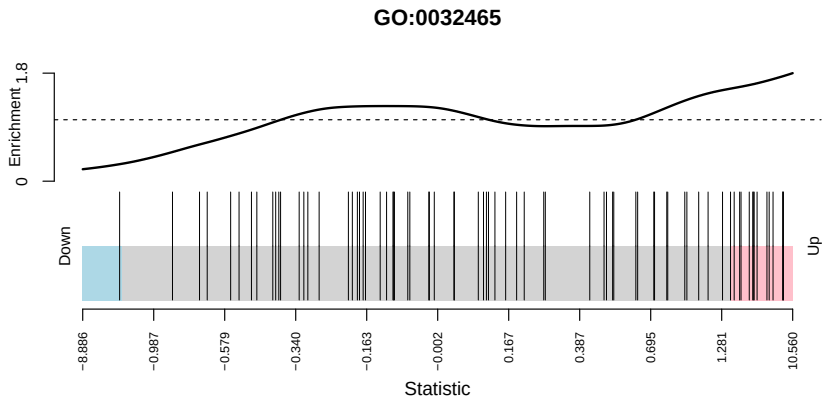
We proceed to run ROAST on the defined gene sets for the contrast of interest. Suppose the comparison of interest is between the virgin and lactating groups in the basal population. We use `fry` to test for multiple gene sets.

```
> con <- makeContrasts(B.virgin-B.lactate, levels=design)
> fr <- fry(y, index=ind, design=design, contrast=con)
> fr
           NGenes Direction  PValue     FDR PValue.Mixed FDR.Mixed
GO:0032465     75        Up 0.00231 0.00364     8.60e-06  1.08e-05
GO:0000281     80        Up 0.00364 0.00364     1.08e-05  1.08e-05
```

Each row corresponds to a single gene set, i.e., GO term. The `NGenes` column gives the number of genes in each set. The net direction of change is determined from the significance of changes in each direction, and is shown in the `Direction` column. The `PValue` provides evidence for whether the majority of genes in the set are DE in the specified direction, whereas the `PValue.Mixed` tests for differential expression in any direction. FDRs are computed from the corresponding p-values across all sets.

A barcode plot can be produced with the `barcodeplot` function to visualize the results for any particular set. In this case, visualization is performed for the gene set defined by GO:0032465. Here, genes are represented by bars and are ranked from left to right by increasing log-fold change. This forms the barcode-like pattern. The line above the barcode shows the relative local enrichment of the vertical bars in each part of the plot. This particular plot suggests that most genes in this set are up-regulated in the virgin group compared to the lactating group.

```
> res <- glmQLFTest(fit, contrast=con)
> barcodeplot(res$table$logFC, ind[[1]], main=names(ind)[1])
```



4.4.12 Setup

This analysis was conducted on:

```
> sessionInfo()

R version 4.4.0 (2024-04-24 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 10 x64 (build 19045)

Matrix products: default


locale:
[1] LC_COLLATE=English_Australia.utf8  LC_CTYPE=English_Australia.utf8
[3] LC_MONETARY=English_Australia.utf8 LC_NUMERIC=C
[5] LC_TIME=English_Australia.utf8

time zone: Australia/Sydney
tzcode source: internal

attached base packages:
[1] stats4    stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] GO.db_3.19.1         org.Mm.eg.db_3.19.1  AnnotationDbi_1.65.2
 [4] IRanges_2.37.1       S4Vectors_0.41.7     Biobase_2.63.1
 [7] BiocGenerics_0.49.1  knitr_1.46           BiocStyle_2.31.0
[10] edgeR_4.1.32         limma_3.59.10

loaded via a namespace (and not attached):
 [1] bit_4.0.5               jsonlite_1.8.8          compiler_4.4.0
 [4] BiocManager_1.30.22     highr_0.10              crayon_1.5.2
 [7] Rcpp_1.0.12             blob_1.2.4              Biostrings_2.71.6
[10] splines_4.4.0           png_0.1-8               yaml_2.3.8
[13] fastmap_1.1.1           statmod_1.5.0           lattice_0.22-6
[16] R6_2.5.1                XVector_0.43.1          GenomeInfoDb_1.39.14
[19] GenomeInfoDbData_1.2.12 DBI_1.2.2               rlang_1.1.3
[22] KEGGREST_1.43.0         cachem_1.0.8            xfun_0.43
[25] bit64_4.0.5             RSQLite_2.3.6           memoise_2.0.1
[28] cli_3.6.2               zlibbioc_1.49.3         digest_0.6.35
[31] grid_4.4.0              locfit_1.5-9.9          vctrs_0.6.5
[34] evaluate_0.23           rmarkdown_2.26          httr_1.4.7
[37] pkgconfig_2.0.3         UCSC.utils_0.99.7       tools_4.4.0
[40] htmltools_0.5.8.1
```

4.5 Differential splicing analysis of Foxp1-deficient mice

4.5.1 Introduction

The RNA-Seq data of this case study was from Fu *et al* [16]. This study concerned the role of Foxp1 deletion in cellular differentiation and development in mouse mammary gland. RNA-seq profiles of the basal and luminal cell populations of mice with Foxp1 knock-out and control were generated. In particular, the following two exons of the Foxp1 gene were silenced in the knock-out samples [16].

No.	Exon	Start	End	Length
14	ENSMUSE00000499236	98,945,426	98,945,347	80
15	ENSMUSE00001069281	98,944,720	98,944,619	102

The RNA-Seq data of the 12 samples are available on the GEO repository as series **GSE118617**. Each combination of cell type (basal and luminal) and genotype (control and Foxp1 knock-out) comprises of three biological samples. The details of the samples are shown in the target file below.

```
> targets <- read.delim("targets.txt", header=TRUE)
```

```
> targets
      Samples        GSM        SRR                 Description
1  Basal-CT-1 GSM3335607 SRR7701128          Basal Control Rep1
2  Basal-KO-1 GSM3335608 SRR7701129   Basal Foxp1-knockout Rep1
3  Basal-CT-2 GSM3335609 SRR7701130          Basal Control Rep2
4  Basal-KO-2 GSM3335610 SRR7701131   Basal Foxp1-knockout Rep2
5  Basal-CT-3 GSM3335611 SRR7701132          Basal Control Rep3
6  Basal-KO-3 GSM3335612 SRR7701133   Basal Foxp1-knockout Rep3
7    Lum-CT-1 GSM3335613 SRR7701134        Luminal Control Rep1
8    Lum-KO-1 GSM3335614 SRR7701135 Luminal Foxp1-knockout Rep1
9    Lum-CT-2 GSM3335615 SRR7701136        Luminal Control Rep2
10   Lum-KO-2 GSM3335616 SRR7701137 Luminal Foxp1-knockout Rep2
11   Lum-CT-3 GSM3335617 SRR7701138        Luminal Control Rep3
12   Lum-KO-3 GSM3335618 SRR7701139 Luminal Foxp1-knockout Rep3
```

4.5.2 Read alignment and processing

We use Rsubread[24] for read alignment and count quantification. The FASTQ files of the 12 samples were first downloaded using the SRA Toolkit. Then an index file of the mouse reference genome (GRCm39) was built in Rsubread using the FASTA files downloaded from the GENCODE database https://www.gencodegenes.org/mouse/. For more details of building index, please check out the Rsubread user's guide.

Assuming the index of the mouse genome (mm39) is already available, we performed read alignment as follows.

```
> library(Rsubread)
> file <- dir(pattern="*.fastq.gz")
```

```
> bam <- gsub(".fastq.gz$", ".bam", file)
> align(index="mm39", readfile1=file, input_format="gzFASTQ", output_file=bam)
```

Next we counted the number of reads overlapping each annotated exon of each gene. We use `featureCounts` in Rsubread with the mouse mm39 inbuilt annotation.

```
> Foxp1 <- featureCounts(bam, isPairedEnd=FALSE, annot.inbuilt="mm39",
+     useMetaFeatures=FALSE, allowMultiOverlap=TRUE)
```

4.5.3 Count loading and annotation

We create a `DGEList` object as follows

```
> library(edgeR)
> y <- DGEList(counts=Foxp1$counts, genes=Foxp1$annotation)
> dim(y)
[1] 285931     12
> head(y$genes)
     GeneID  Chr    Start      End Strand Length
1 100009600 chr9 20973689 20974013      -    325
2 100009600 chr9 20974190 20974283      -     94
3 100009600 chr9 20974610 20974692      -     83
4 100009600 chr9 20977320 20977673      -    354
5 100009600 chr9 20978236 20978389      -    154
6 100009609 chr7 84589377 84590296      -    920
```

The annotation includes Entrez ID and the length, chromosome and start and stop position of each exon. Mouse gene symbols were added to the exon annotation, using the annotation in the `org.Mm.eg.db` package.

```
> require(org.Mm.eg.db)
> Symbol <- mapIds(org.Mm.eg.db, keys=rownames(y), keytype="ENTREZID",
+     column="SYMBOL")
> y$genes$Symbol <- Symbol
> head(y$genes)
     GeneID  Chr    Start      End Strand Length  Symbol
1 100009600 chr9 20973689 20974013      -    325   Zglp1
2 100009600 chr9 20974190 20974283      -     94   Zglp1
3 100009600 chr9 20974610 20974692      -     83   Zglp1
4 100009600 chr9 20977320 20977673      -    354   Zglp1
5 100009600 chr9 20978236 20978389      -    154   Zglp1
6 100009609 chr7 84589377 84590296      -    920 Vmn2r65
```

4.5.4 Filtering and normalization

The lowly expressed exons were filtered out prior to the downstream analysis.

```
> group <- gsub("-[1-3]$", "", colnames(y))
> group <- factor(gsub("-", "_", group))
```

```
> y$samples$group <- group
> keep <- filterByExpr(y, group=group)
> table(keep)

keep
 FALSE   TRUE
179116 106815

> y <- y[keep, , keep.lib.sizes=FALSE]
```

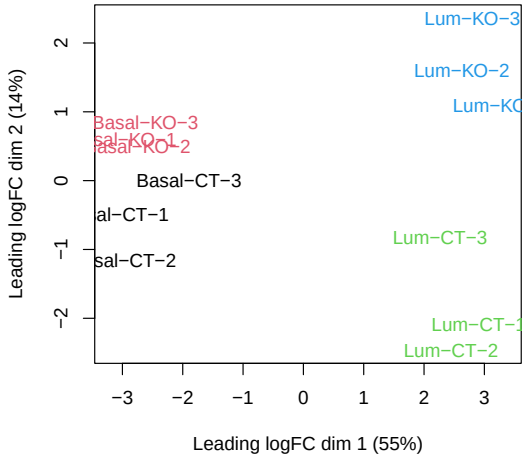
TMM normalization is performed to eliminate composition biases between libraries.

```
> y <- normLibSizes(y)
> y$samples

              group lib.size norm.factors
Basal-CT-1 Basal_CT 10679016        0.982
Basal-KO-1 Basal_KO 11291378        0.895
Basal-CT-2 Basal_CT 11731453        1.014
Basal-KO-2 Basal_KO 11164225        1.025
Basal-CT-3 Basal_CT 10846137        1.055
Basal-KO-3 Basal_KO 10965059        0.941
Lum-CT-1     Lum_CT 11537126        0.977
Lum-KO-1     Lum_KO 10560091        0.999
Lum-CT-2     Lum_CT 10236505        1.108
Lum-KO-2     Lum_KO 10577400        1.076
Lum-CT-3     Lum_CT 10672170        0.974
Lum-KO-3     Lum_KO 16388823        0.972
```

4.5.5 Data exploration

The data can be explored by generating multi-dimensional scaling (MDS) plots. This visualizes the differences between the expression profiles of different samples in two dimensions.

```
> plotMDS(y, col=c(1:4)[group])
```



The MDS plot shows the basal and luminal samples are well separated in the first dimension, whereas the control and Foxp1 knock-out samples are separated in the second dimension.

4.5.6 Design matrix

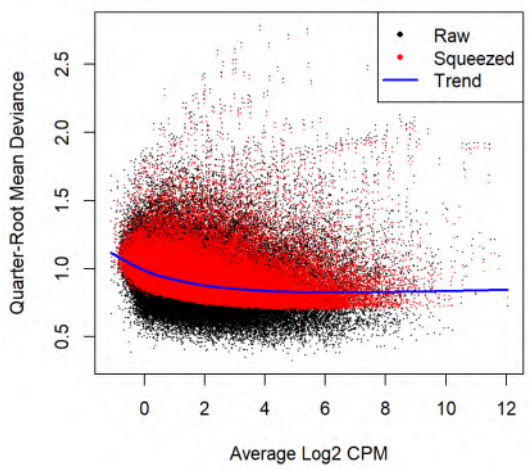
Since there are four groups of samples in this RNA-seq experiment, a design matrix is created as follows:

```
> design <- model.matrix(~ 0 + group)
> colnames(design) <- gsub("group", "", colnames(design))
> design
   Basal_CT Basal_KO Lum_CT Lum_KO
1         1        0      0      0
2         0        1      0      0
3         1        0      0      0
4         0        1      0      0
5         1        0      0      0
6         0        1      0      0
7         0        0      1      0
8         0        0      0      1
9         0        0      1      0
10        0        0      0      1
11        0        0      1      0
12        0        0      0      1
attr(,"assign")
[1] 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$group
[1] "contr.treatment"
```

4.5.7 Dispersion estimation

Under the current QL pipeline, technical and biological variation can be estimated simultaneously in the `glmQLFit` function. The QL dispersion estimates can be visualized with the `plotQLDisp` function.

```
> fit <- glmQLFit(y, design, robust=TRUE)
> plotQLDisp(fit)
```



4.5.8 Differential expression

We test for differentially expressed exons between the Foxp1-KO and the control in the luminal population using the QL F-test. The contrast of interest can be constructed using the `makeContrasts` function.

```
> contr <- makeContrasts(Lum_KO - Lum_CT, levels=design)
> qlf <- glmQLFTest(fit, contrast=contr)
```

The top set of most significant exons can be examined with `topTags`. Here, a positive log-fold change represents exons that are up in Foxp1-KO over control. Multiplicity correction is performed by applying the Benjamini-Hochberg method on the p-values, to control the false discovery rate (FDR).

```
> topTags(qlf)

Coefficient:  -1*Lum_CT 1*Lum_KO
         GeneID   Chr     Start       End Strand Length Symbol logFC logCPM
50518.1   50518  chr2 154887532 154887701      +    170      a  9.98   3.38
12837     12837 chr16  57444619  57449180      -   4562 Col8a1 -5.09   4.62
12990.8   12990  chr5  87830048  87830437      +    390 Csn1s1 -5.67   2.07
50518     50518  chr2 154855490 154855561      +     72      a  7.22   2.52
12990.7   12990  chr5  87828679  87828821      +    143 Csn1s1 -7.89   1.28
112422   112422  chr4 147696393 147698505      -   2113 Zfp979 -5.64   1.13
494504.4 494504 chr18  63084901  63086886      +   1986 Apcdd1  3.88   4.68
```

```
494504.2 494504 chr18  63069977  63070508      +    532 Apcdd1  4.23   2.53
50518.3   50518  chr2 154892548 154892932      +    385      a  7.23   4.04
27220.1   27220 chr13 100036465 100036587      -    123 Cartpt  7.84   1.37
             F   PValue      FDR
50518.1  283.1 6.82e-12 5.69e-07
12837    315.5 1.06e-11 5.69e-07
12990.8  179.0 5.36e-11 1.91e-06
50518    135.7 1.39e-10 3.19e-06
12990.7  134.2 1.59e-10 3.19e-06
112422   196.3 1.79e-10 3.19e-06
494504.4 188.9 4.46e-10 6.80e-06
494504.2 172.6 8.96e-10 1.20e-05
50518.3  123.7 3.50e-09 4.03e-05
27220.1   83.7 4.39e-09 4.03e-05
```

The total number of DE exons in each direction at a FDR of 5% can be examined with `decideTests`.

```
> is.de <- decideTests(qlf, p.value=0.05)
> summary(is.de)

       -1*Lum_CT 1*Lum_KO
Down                 300
NotSig            105230
Up                  1285
```

4.5.9 Alternative splicing

We detect alternative splicing by testing for differential exon usage between Foxp1-KO and control in the luminal population.

```
> sp <- diffSpliceDGE(fit, contrast=contr, geneid="GeneID", exonid="Start")

Total number of exons:  106815
Total number of genes:  14816
Number of genes with 1 exon:  3646
Mean number of exons in a gene:  7
Max number of exons in a gene:  106
```

The top differentially used exons are shown below:

```
> topSpliceDGE(sp, test="exon")

       GeneID   Chr     Start       End Strand Length   Symbol  logFC exon.F
49046  108655  chr6  98921580  98921681      -    102    Foxp1 -2.665  121.9
49047  108655  chr6  98922308  98922387      -     80    Foxp1 -3.056  120.4
122564 208263  chr1 155895714 155895769      -     56 Tor1aip1  5.000   87.4
140297  22228  chr7 100145293 100145414      +    122     Ucp2 -1.493   38.0
100766  17758  chr9 109831814 109834852      +   3039     Map4 -1.279   32.9
184802 319448 chr14  72937689  72941443      -   3755   Fndc3a -1.380   29.6
264648  74383  chr3  89906896  89907560      -    665   Ubap2l  0.918   27.0
9807   102436  chr9 123283775 123283944      +    170    Lars2  2.509   29.6
9548   102103  chr8  41494510  41494968      -    459    Mtus1  1.664   28.4
```

```
9791   102436  chr9 123200920 123201168      +    249    Lars2  2.324   28.7
        P.Value      FDR
49046  1.35e-21 1.07e-16
49047  2.08e-21 1.07e-16
122564 6.71e-15 2.31e-10
140297 2.89e-08 7.44e-04
100766 4.18e-08 8.62e-04
184802 1.39e-07 2.38e-03
264648 4.42e-07 6.52e-03
9807   6.18e-07 7.85e-03
9548   6.85e-07 7.85e-03
9791   8.47e-07 8.74e-03
```

The successful elimination of the two targeted exons is demonstrated by the fact that the two specific exons of the Foxp1 gene rank as the top two exons with the greatest difference in usage.

Two different methods can be used to examine the differential splicing results at the gene level: the Simes' method and the gene-level F-test. The Simes' method is likely to be more powerful when only a minority of the exons for a gene are differentially spliced. The F-tests are likely to be powerful for genes in which several exons are differentially spliced.

The top spliced genes under the Simes' method are shown below:

```
> topSpliceDGE(sp, test="Simes")

       GeneID   Chr Strand   Symbol NExons  P.Value      FDR
49079  108655  chr6      -    Foxp1     20 1.97e-20 2.20e-16
122571 208263  chr1      - Tor1aip1     11 6.71e-14 3.75e-10
140303  22228  chr7      +     Ucp2      9 2.31e-07 8.59e-04
100783  17758  chr9      +     Map4     21 8.35e-07 2.33e-03
9811   102436  chr9      +    Lars2      9 3.39e-06 6.71e-03
184804 319448 chr14      -   Fndc3a     27 3.61e-06 6.71e-03
9555   102103  chr8      -    Mtus1     11 6.85e-06 1.09e-02
264677  74383  chr3      -   Ubap2l     29 1.24e-05 1.73e-02
255693  72181  chr4      -    Nsun4      7 2.41e-05 2.99e-02
114280 194401  chr6      -   Mical3     28 3.20e-05 3.57e-02
```

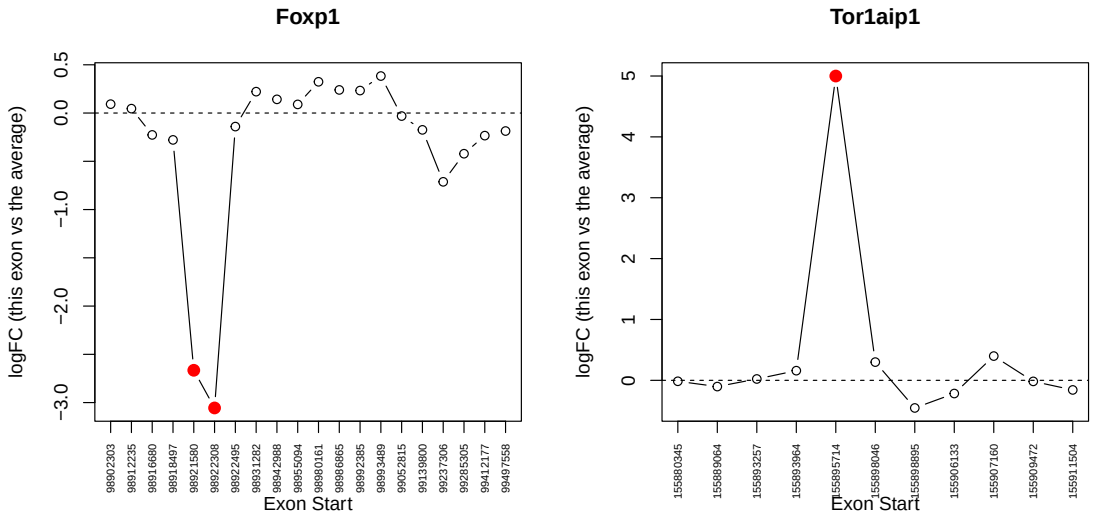
The top spliced genes identified by F-tests are shown below:

```
> topSpliceDGE(sp, test="gene")

       GeneID   Chr Strand   Symbol NExons gene.F  P.Value      FDR
49079  108655  chr6      -    Foxp1     20  14.35 5.02e-26 5.61e-22
9811   102436  chr9      +    Lars2      9  20.00 5.64e-16 3.15e-12
122571 208263  chr1      - Tor1aip1     11   9.88 5.83e-11 2.17e-07
216231  57738 chr16      -  Slc15a2     20   4.49 3.96e-08 1.11e-04
69092   12095  chr3      -   Bglap3      6   9.67 1.20e-06 2.67e-03
9555   102103  chr8      -    Mtus1     11   5.48 2.33e-06 4.34e-03
127951 213006  chr1      -   Mfsd4a      4  10.23 2.87e-05 4.59e-02
140303  22228  chr7      +     Ucp2      9   4.80 7.99e-05 1.12e-01
255693  72181  chr4      -    Nsun4      7   5.49 1.42e-04 1.76e-01
148866 227746  chr2      -   Rabepk      9   4.38 2.15e-04 2.41e-01
```

As expected, the Foxp1 gene appears as the top gene under both gene-level tests.

We plot all the exons for the top two most differentially spliced genes. Exons that are individually significant are highlighted.

```
> par(mfrow=c(1,2))
> plotSpliceDGE(sp, geneid="Foxp1", genecol="Symbol")
> plotSpliceDGE(sp, geneid="Tor1aip1", genecol="Symbol")
```



We can see that the two Foxp1 exons with start positions of 98921580 and 98922308 are significantly down in the Foxp1 KO samples compared to the control.

4.5.10 Setup

This analysis was conducted on:

```
> sessionInfo()

R version 4.4.0 (2024-04-24 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 10 x64 (build 19045)

Matrix products: default


locale:
[1] LC_COLLATE=English_Australia.utf8  LC_CTYPE=English_Australia.utf8
[3] LC_MONETARY=English_Australia.utf8 LC_NUMERIC=C
[5] LC_TIME=English_Australia.utf8

time zone: Australia/Sydney
tzcode source: internal

attached base packages:
```

```
[1] stats4    stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] org.Mm.eg.db_3.19.1  AnnotationDbi_1.65.2 IRanges_2.37.1
 [4] S4Vectors_0.41.7     Biobase_2.63.1       BiocGenerics_0.49.1
 [7] edgeR_4.1.28         limma_3.59.10        knitr_1.46
[10] BiocStyle_2.31.0

loaded via a namespace (and not attached):
 [1] bit_4.0.5               jsonlite_1.8.8          compiler_4.4.0
 [4] BiocManager_1.30.22     highr_0.10              crayon_1.5.2
 [7] Rcpp_1.0.12             blob_1.2.4              Biostrings_2.71.6
[10] splines_4.4.0           png_0.1-8               yaml_2.3.8
[13] fastmap_1.1.1           statmod_1.5.0           lattice_0.22-6
[16] R6_2.5.1                XVector_0.43.1          GenomeInfoDb_1.39.14
[19] GenomeInfoDbData_1.2.12 DBI_1.2.2               rlang_1.1.3
[22] KEGGREST_1.43.0         cachem_1.0.8            xfun_0.43
[25] bit64_4.0.5             RSQLite_2.3.6           memoise_2.0.1
[28] cli_3.6.2               zlibbioc_1.49.3         digest_0.6.35
[31] grid_4.4.0              locfit_1.5-9.9          vctrs_0.6.5
[34] evaluate_0.23           rmarkdown_2.26          httr_1.4.7
[37] pkgconfig_2.0.3         UCSC.utils_0.99.7       tools_4.4.0
[40] htmltools_0.5.8.1
```

4.6 Differential transcript expression of human lung adenocarcinoma cell lines

4.6.1 Introduction

We use the RNA-Seq data of the human lung adenocarninoma cell lines experiment from Dong *et al* [12] as a case study for a differential expression analysis at the transcript-level with edgeR . The DTE analysis focuses on the Illumina™short read RNA-Seq samples of cell lines NCI-H1975 and HCC827, for which three biological replicates were sequenced with paired-end sequencing protocol. Data for all six samples are available on the GEO repository as series **GSE172421**. The details of the samples are shown in the target file below.

```
> targets <- read.delim("targets.txt", header=TRUE)
```

```
> targets

   Group Replicate   Sample        GSM
1  H1975         1  H1975.1 GSM5255695
2  H1975         2  H1975.2 GSM5255696
3  H1975         3  H1975.3 GSM5255697
4 HCC827         1 HCC827.1 GSM5255704
5 HCC827         2 HCC827.2 GSM5255705
6 HCC827         3 HCC827.3 GSM5255706
```

4.6.2 Read pseudoalignment and processing

We use Salmon[34] for pseudoalignment and quantification of sequence reads at the transcript-level. The FASTQ files of the six samples were first downloaded using the SRA Toolkit. Then, a transcriptome index based on the human GENCODE annotation (version 33) for the human reference genome build hg38 was created with Salmon and used during read quantification. For more details on the creation of the transcriptome index and read pseudoalignment, please consult the Salmon's documentation.

Assuming the quantification files from Salmon have been generated, we use `catchSalmon` to import transcript-level counts and estimate the associated mapping ambiguity overdispersion [1].

```
> library(edgeR)
> quant <- dirname(list.files(".","quant.sf",recursive = TRUE,full.names = TRUE))
> catch <- catchSalmon(paths = quant)
```

4.6.3 Count loading and annotation

To account for the mapping ambiguity resulting from the assignment of reads to transcripts, we divide the transcript-level counts by the estimated mapping ambiguity overdispersion associated to each transcript. Information about the samples are brought in from the targets file. Then, a `DGEList` object can be created as follows.

```
> scaled.counts <- catch$counts/catch$annotation$Overdispersion
> samples <- targets[match(colnames(scaled.counts),targets$GSM),]
> y <- DGEList(counts = scaled.counts,
+              samples = samples,
+              group = samples$Group,
+              genes = catch$annotation)
> dim(y)

[1] 227063      6

> head(y$genes)

                  Length EffectiveLength Overdispersion
ENST00000456328.2   1657            1466           4.85
ENST00000450305.2    632             441           2.16
ENST00000488147.1   1351            1160           3.40
ENST00000619216.1     68              68           2.16
ENST00000473358.1    712             521           1.00
ENST00000469289.1    535             345           2.16
```

For the Ensembl-oriented GENCODE annotation, information about the transcripts can be easily obtained from the `AnnotationHub` package and added to the `DGEList` object. Transcript information might include the transcript biotype as well as the associated gene symbol. The relevant `AnnotationHub` ID for the specific GENCODE annotation version 33 used in this case study is `AH78783` (Ensembl annotation version 99) and used below.

```
> require(AnnotationHub)
> ah <- AnnotationHub()
> edb <- ah[['AH78783']]
> edb.info <- select(edb,keys(edb),c("TXIDVERSION","TXBIOTYPE","SYMBOL"))
```

```
> edb.info <- edb.info[match(rownames(y),edb.info$TXIDVERSION),]
>
> y$genes <- cbind(y$genes,edb.info[,-c(1,2)])
> head(y$genes)
```

```
                  Length EffectiveLength Overdispersion
ENST00000456328.2   1657            1466           4.85
ENST00000450305.2    632             441           2.16
ENST00000488147.1   1351            1160           3.40
ENST00000619216.1     68              68           2.16
ENST00000473358.1    712             521           1.00
ENST00000469289.1    535             345           2.16
                                      TXBIOTYPE      SYMBOL
ENST00000456328.2               processed_transcript     DDX11L1
ENST00000450305.2 transcribed_unprocessed_pseudogene     DDX11L1
ENST00000488147.1             unprocessed_pseudogene      WASH7P
ENST00000619216.1                              miRNA   MIR6859-1
ENST00000473358.1                            lncRNA MIR1302-2HG
ENST00000469289.1                            lncRNA MIR1302-2HG
```

4.6.4 Filtering and normalization

Lowly expressed transcripts are filtered out prior to the downstream analysis.

```
> keep <- filterByExpr(y)
> table(keep)
```

```
keep
 FALSE   TRUE
196325  30738
```

```
> y <- y[keep, , keep.lib.sizes=FALSE]
```

After scaling counts by the estimated mapping ambiguity overdispersion, scaling factors can computed using the TMM method to convert the resulting library sizes to effective library sizes.

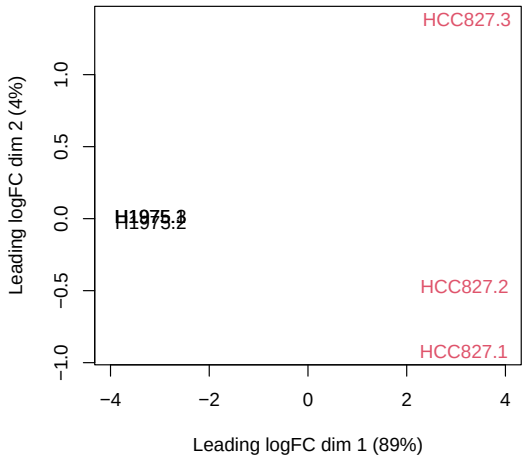
```
> y <- normLibSizes(y)
> y$samples
```

```
            group lib.size norm.factors  Group Replicate   Sample        GSM
GSM5255695  H1975 15361370        1.140  H1975         1  H1975.1 GSM5255695
GSM5255696  H1975 16622853        1.160  H1975         2  H1975.2 GSM5255696
GSM5255697  H1975 13842625        1.159  H1975         3  H1975.3 GSM5255697
GSM5255704 HCC827 15911639        0.916 HCC827         1 HCC827.1 GSM5255704
GSM5255705 HCC827 67664442        0.814 HCC827         2 HCC827.2 GSM5255705
GSM5255706 HCC827 14668555        0.875 HCC827         3 HCC827.3 GSM5255706
```

4.6.5 Data exploration

Analogous to a DGE analysis, MDS plots can also be used to visualize differences between the expression profiles of different samples with transcript-level counts.

```
> plotMDS(y,col = c(1:2)[y$samples$group],labels = y$samples$Sample,xlim = c(-4,4))
```



The MDS plot shows that cell lines are well separated in the first dimension.

4.6.6 Design matrix

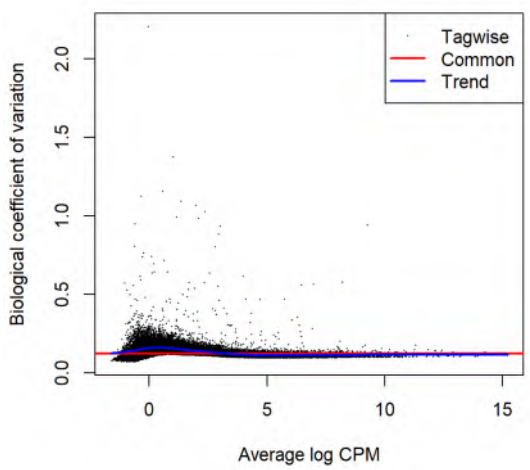
The design matrix used to assess DTE between cell lines NCI-H1975 and HCC827 is then created below.

```
> design <- model.matrix(~ 0 + group,data = y$samples)
> colnames(design) <- gsub("group", "", colnames(design))
> design

           H1975 HCC827
GSM5255695     1      0
GSM5255696     1      0
GSM5255697     1      0
GSM5255704     0      1
GSM5255705     0      1
GSM5255706     0      1
attr(,"assign")
[1] 1 1
attr(,"contrasts")
attr(,"contrasts")$group
[1] "contr.treatment"
```

4.6.7 Dispersion estimation

We estimate NB dispersions using the `estimateDisp` function and visualize the estimated values with `plotBCV`.

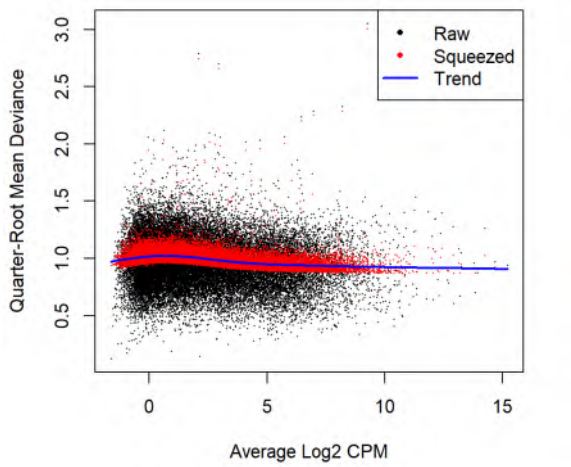
```
> y <- estimateDisp(y, design, robust=TRUE)
> y$common.dispersion
[1] 0.0147
> plotBCV(y)
```



Note that this NB dispersion estimation step is now optional as all the NB dispersion estimates will not be used further under the latest quasi-likelihood (QL) pipeline.

For DTE analyses, we still recommend using the quasi-likelihood (QL) pipeline for stricter error rate control by accounting for the uncertainty associated with the dispersion estimation. To this end, we estimate QL dispersions via `glmQLFit` and visualize the estimated values with `plotQLDisp`.

```
> fit <- glmQLFit(y, design, robust=TRUE)
> plotQLDisp(fit)
```



4.6.8 Differential expression

Differentially expressed transcripts are tested between cell lines NCI-H1975 and HCC827 using the QL F-test. We use the `makeContrasts` function to create the relevant contrast.

```
> contr <- makeContrasts(HCC827 - H1975, levels=design)
> qlf <- glmQLFTest(fit, contrast=contr)
```

The total number of DE transcripts up- and down-regulated at a FDR of 5% can be examined with `decideTests`. We use the default Benjamini-Hochberg method to adjust p-values for multiple testing.

```
> is.de <- decideTests(qlf, p.value=0.05)
> summary(is.de)

       -1*H1975 1*HCC827
Down                11049
NotSig               9464
Up                  10225
```

The top set of most significant differentially expressed transcripts can be examined with `topTags`, with a positive log-fold change representing up-regulation in expression levels in HCC827 over H1975.

```
> tt <- topTags(qlf,n = Inf)
> head(tt)

Coefficient:  -1*H1975 1*HCC827
                  Length EffectiveLength Overdispersion            TXBIOTYPE
ENST00000306061.10    704             513           1.18       protein_coding
ENST00000488803.1     879             688           1.28 processed_pseudogene
ENST00000223095.5    3156            2965           1.00       protein_coding
ENST00000372431.8    1889            1698           5.43       protein_coding
ENST00000311852.11   3701            3510           1.66       protein_coding
```

```
ENST00000511685.6   10923           10732           1.34       protein_coding
                     SYMBOL  logFC logCPM    F   PValue      FDR
ENST00000306061.10     MT1E -12.09   6.66 2425 2.13e-23 6.54e-19
ENST00000488803.1     RPS2P5  12.43   8.15 2913 3.40e-22 5.22e-18
ENST00000223095.5  SERPINE1  -8.45   8.42 2959 1.00e-21 1.03e-17
ENST00000372431.8       PLTP -11.27   5.12 1629 4.64e-21 3.09e-17
ENST00000311852.11    MMP14  -8.84   6.85 2496 5.02e-21 3.09e-17
ENST00000511685.6      TENM3 -10.29   4.35 1255 3.12e-20 1.42e-16
```

We can see that the top set of most significant DE transcripts includes several transcripts with associated protein-coding biotype.

In addition, our DE analysis at the transcript-level reveals several interesting genes with contrasting up- and down-regulation of their associated transcripts between cell lines. Such genes include the BCL2L1 gene, for which one of its protein-coding transcripts (ENST00000376062.6) is down-regulated in contrast to the remaining up-regulated transcripts in the HCC827 cell line.

```
> tt$table[tt$table$SYMBOL == 'BCL2L1',]

                  Length EffectiveLength Overdispersion      TXBIOTYPE SYMBOL
ENST00000307677.5   2574            2383           5.15 protein_coding BCL2L1
ENST00000456404.5    914             723           3.30 protein_coding BCL2L1
ENST00000376062.6   2578            2387           9.67 protein_coding BCL2L1
ENST00000422920.1   1157             966           2.17 protein_coding BCL2L1
ENST00000376055.8   2383            2192          10.14 protein_coding BCL2L1
                    logFC  logCPM     F   PValue      FDR
ENST00000307677.5  1.293  6.0551 82.01 3.43e-08 2.30e-07
ENST00000456404.5  2.934 -0.0296 35.24 1.20e-05 3.76e-05
ENST00000376062.6 -0.798  3.6044 25.35 8.16e-05 2.10e-04
ENST00000422920.1  0.244  3.5633  2.51 1.30e-01 1.67e-01
ENST00000376055.8  0.320 -0.0820  1.06 3.17e-01 3.70e-01
```

4.6.9 Setup

This analysis was conducted on:

```
> sessionInfo()

R version 4.4.0 (2024-04-24 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 10 x64 (build 19045)

Matrix products: default


locale:
[1] LC_COLLATE=English_Australia.utf8  LC_CTYPE=English_Australia.utf8
[3] LC_MONETARY=English_Australia.utf8 LC_NUMERIC=C
[5] LC_TIME=English_Australia.utf8

time zone: Australia/Sydney
```

```
tzcode source: internal

attached base packages:
[1] stats4    stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] ensembldb_2.27.1        AnnotationFilter_1.27.0 GenomicFeatures_1.55.4
 [4] AnnotationDbi_1.65.2    Biobase_2.63.1          GenomicRanges_1.55.4
 [7] GenomeInfoDb_1.39.14    IRanges_2.37.1          S4Vectors_0.41.7
[10] AnnotationHub_3.11.5    BiocFileCache_2.11.2    dbplyr_2.5.0
[13] BiocGenerics_0.49.1     edgeR_4.1.28            limma_3.59.10
[16] knitr_1.46              BiocStyle_2.31.0

loaded via a namespace (and not attached):
 [1] tidyselect_1.2.1            dplyr_1.1.4
 [3] blob_1.2.4                  filelock_1.0.3
 [5] Biostrings_2.71.6           bitops_1.0-7
 [7] lazyeval_0.2.2              fastmap_1.1.1
 [9] RCurl_1.98-1.14             GenomicAlignments_1.39.5
[11] XML_3.99-0.16.1             digest_0.6.35
[13] mime_0.12                   lifecycle_1.0.4
[15] ProtGenerics_1.35.4         statmod_1.5.0
[17] KEGGREST_1.43.0             RSQLite_2.3.6
[19] magrittr_2.0.3              compiler_4.4.0
[21] rlang_1.1.3                 tools_4.4.0
[23] utf8_1.2.4                  yaml_2.3.8
[25] rtracklayer_1.63.3          S4Arrays_1.3.7
[27] bit_4.0.5                   curl_5.2.1
[29] DelayedArray_0.29.9         abind_1.4-5
[31] BiocParallel_1.37.1         withr_3.0.0
[33] purrr_1.0.2                 grid_4.4.0
[35] fansi_1.0.6                 SummarizedExperiment_1.33.3
[37] cli_3.6.2                   rmarkdown_2.26
[39] crayon_1.5.2                generics_0.1.3
[41] httr_1.4.7                  rjson_0.2.21
[43] DBI_1.2.2                   cachem_1.0.8
[45] splines_4.4.0               zlibbioc_1.49.3
[47] parallel_4.4.0              BiocManager_1.30.22
[49] XVector_0.43.1              restfulr_0.0.15
[51] matrixStats_1.3.0           vctrs_0.6.5
[53] Matrix_1.7-0                jsonlite_1.8.8
[55] bit64_4.0.5                 locfit_1.5-9.9
[57] glue_1.7.0                  codetools_0.2-20
[59] BiocVersion_3.19.1          BiocIO_1.13.0
[61] UCSC.utils_0.99.7           tibble_3.2.1
[63] pillar_1.9.0                rappdirs_0.3.3
[65] htmltools_0.5.8.1           GenomeInfoDbData_1.2.12
[67] R6_2.5.1                    evaluate_0.23
[69] lattice_0.22-6              highr_0.10
[71] png_0.1-8                   Rsamtools_2.19.4
```

```
[73] memoise_2.0.1              Rcpp_1.0.12
[75] SparseArray_1.3.5          xfun_0.43
[77] MatrixGenerics_1.15.1      pkgconfig_2.0.3
```

4.7 CRISPR-Cas9 knockout screen analysis

4.7.1 Introduction

Dai *et al.* (2014) [11] describe the use of `edgeR` to analyze data from pooled genetic screens utilizing either shRNAs or CRISPR-Cas9 to disrupt gene expression in a population of cells.

In this case study we analyze data from a pooled screen that uses CRISPR-Cas9 (clustered regularly interspaced short palindromic repeats-associated nuclease Cas9) knockout technology. In this example, a library of around 64,000 sgRNAs (as used in Shalem *et al.* 2014 [44]) were screened to look for genes that may lead to resistance from a particular drug. This unpublished data set has been anonymised.

4.7.2 Sequence processing

Multiple single guide RNAs (sgRNAs) per gene (generally between 3-6) were included in the screen. Below we read in the raw sequences from the paired end fastq files *screen4_R1.fastq* and *screen4_R2.fastq* using the `processAmplicons` function in edgeR. This screen employed a dual indexing strategy where the first 8 bases from each pair of reads contained an index sequence that uniquely identifies which sample a particular sgRNA sequence originated from. Matches between sample indexes and sgRNAs listed in the files *Samples4.txt* and *sgRNAs4.txt* are identified by `processAmplicons` to produce a `DGEList` of counts.

```
> library(edgeR)
> sampleanno <- read.table("Samples4.txt", header=TRUE, sep="\t")
> sgseqs <- read.table("sgRNAs4.txt", header=TRUE, sep="\t")
> x <- processAmplicons("screen4_R1.fastq", readfile2="screen4_R2.fastq",
+       barcodefile="Samples4.txt", hairpinfile="sgRNAs4.txt",
+       verbose=TRUE)
```

Note that this dual indexing strategy requires an additional column named *'SequencesRev'* in the file that contains the sample annotation information. Also, `readFile2` must be specified.

The output `DGEList` is available here.

4.7.3 Filtering and data exploration

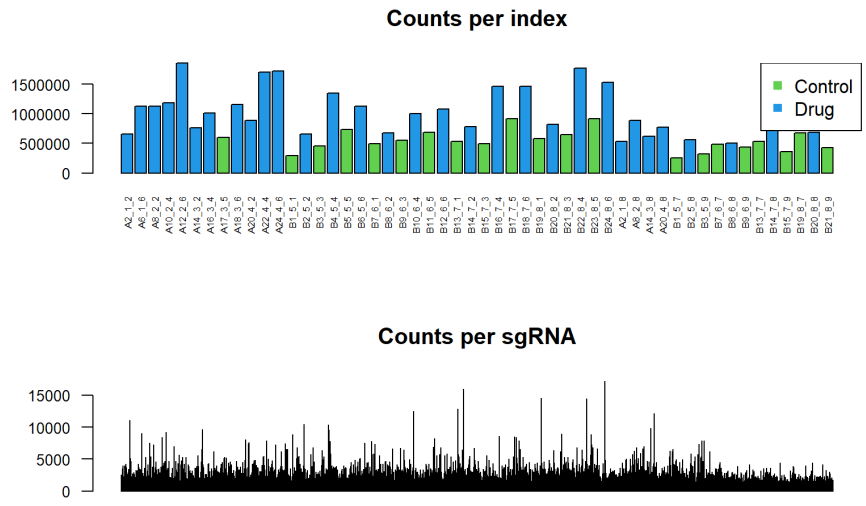
We next filter out sgRNAs and samples with low numbers of reads. Need a CPM greater than 5 in 15 or more samples to keep sgRNAs, and at least 100,000 reads to keep a given sample.

```
> table(x$samples$group)


  Drug NoDrug
    40     32
```

```
> selr <- rowSums(cpm(x$counts)>5)>=15
> selc <- colSums(x$counts)>=100000
> x <- x[selr,selc]
```

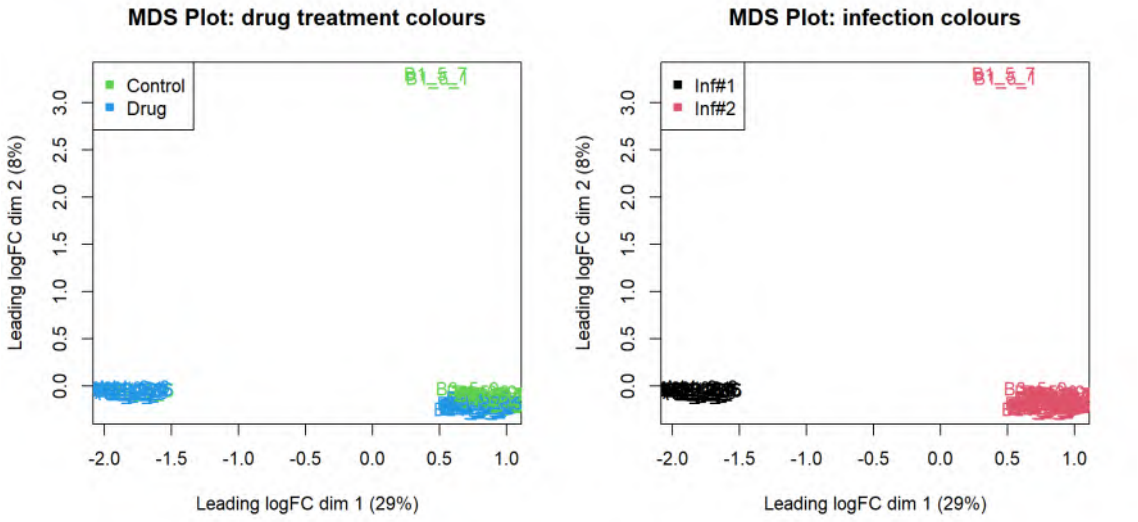
We plot number of sgRNAs that could be matched per sample and total for each sgRNA across all samples .

```
> cols <- as.numeric(x$samples$group)+2
> par(mfrow=c(2,1))
> barplot(colSums(x$counts), las=2, main="Counts per index",
+     col=cols, cex.names=0.5, cex.axis=0.8)
> legend("topright", legend=c("Control", "Drug"), col=c(3,4), pch=15)
> barplot(rowSums(x$counts), las=2, main="Counts per sgRNA",
+     axisnames=FALSE, cex.axis=0.8)
```



A multidimensional scaling plot was generated to assess the consistency between replicate samples. There is a clear separation between the two infections, indicating the need to incorporate an effect for this in the GLM.

```
> cols2 <- x$samples$Infection
> par(mfrow=c(1,2))
> plotMDS(x, col=cols, main="MDS Plot: drug treatment colours")
> legend("topleft", legend=c("Control", "Drug"), col=c(3,4), pch=15)
> plotMDS(x, col=cols2, main="MDS Plot: infection colours")
> legend("topleft", legend=c("Inf#1", "Inf#2"), col=c(1,2), pch=15)
```



4.7.4 Design matrix

A design matrix is set up for the GLM analysis, and the sgRNA-specific variation is estimated and plotted (while taking into account both drug treatment and infection number).

```
> treatment <- relevel(as.factor(x$samples$group), "NoDrug")
> infection <- as.factor(x$samples$Infection)
> des <- model.matrix(~treatment+infection)
> des[1:5,]

  (Intercept) treatmentDrug infection2
1           1             0          0
2           1             0          0
3           1             0          0
4           1             0          0
5           1             0          0

> colnames(des)[2:3] <- c("Drug", "Infection2")
```

4.7.5 Differential representation analysis

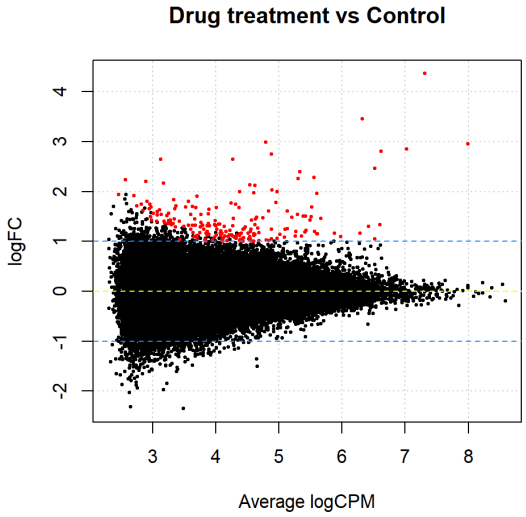
We use the function `glmQLFit` to fit the sgRNA-specific models and `glmQLFTest` to do the testing between the drug treated and control samples. The top ranked sgRNAs are listed using the `topTags` function.

```
> fit <- glmQLFit(x, des)
> qlf <- glmQLFTest(fit, coef=2)
> topTags(qlf)

Coefficient:  Drug
                  ID            Sequences  Gene logFC logCPM    F   PValue
sgRNA4070   sgRNA4070 GTTGTGCTCAGTACTGACTT  1252  2.95   7.99 1064 6.03e-73
sgRNA816     sgRNA816 TCCGAACTCCCCCTTCCCGA   269  4.36   7.31  981 2.02e-70
```

```
sgRNA6351   sgRNA6351 AAAAACGTATCTATTTTTAC  1957  3.45   6.32  598 1.73e-58
sgRNA12880 sgRNA12880 CTGCACCGAAGAGAGCTGCT  3979  2.85   7.02  499 3.32e-51
sgRNA38819 sgRNA38819 TACGTTGTCGGGCGCCGCCA 11531  2.47   6.52  302 8.27e-39
sgRNA52924 sgRNA52924 CCACCGCGTTCCACTTCTTG 16395  2.81   6.62  261 1.47e-35
sgRNA23015 sgRNA23015 CAATTTGATCTCTTCTACTG  6714  2.99   4.80  250 1.27e-34
sgRNA62532 sgRNA62532 AAACACGTCCAGTGCAGCCC 19612  2.75   4.88  245 3.44e-34
sgRNA47062 sgRNA47062 GACCTACCTGCGGAGTCAGA 14328  2.39   5.33  236 3.82e-34
sgRNA3887   sgRNA3887 AACGCTGGACTCGAATGGCC  1194  2.26   5.31  234 3.25e-33
                 FDR
sgRNA4070  3.40e-68
sgRNA816   5.69e-66
sgRNA6351  3.25e-54
sgRNA12880 4.68e-47
sgRNA38819 9.33e-35
sgRNA52924 1.38e-31
sgRNA23015 1.02e-30
sgRNA62532 2.39e-30
sgRNA47062 2.39e-30
sgRNA3887  1.83e-29
```

sgRNAs with FDR < 0.0001 [2] and log-fold-change ≥ 1 are highlighted on a plot of log-fold-change versus log-counts-per-millions by the `plotSmear` function. Since this is a positive screen, we highlight over-represented sgRNAs (i.e. those with positive log-fold-changes) as the model is parameterized to compare drug treatment versus control (coefficient 2 in the design matrix).

```
> thresh <- 0.0001
> lfc <- 1
> top4 <- topTags(qlf, n=Inf)
> top4ids <- top4$table[top4$table$FDR<thresh & top4$table$logFC>=lfc,1]
> plotSmear(qlf, de.tags=top4ids, pch=20, cex=0.6,
+     main="Drug treatment vs Control")

Warning in plot.xy(xy.coords(x, y), type = type, ...):  "panel.first" is not a graphical
parameter

> abline(h=c(-1, 0, 1), col=c("dodgerblue","yellow","dodgerblue"), lty=2)
```



4.7.6 Summarization over multiple sgRNAs targeting the same gene

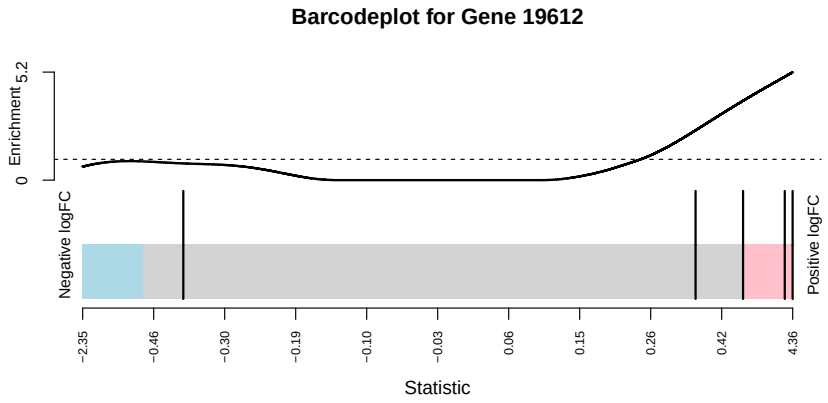
We finish this analysis by summarising data across multiple sgRNAs that target the same gene in order to get a gene-by-gene ranking, rather than a sgRNA-specific one. The *camera* gene-set test [50] is used for this purpose. For this analysis, the collection of sgRNAs that target a specific gene can be regarded as a 'set'. In the code below, we restrict our analysis to genes with more than 3 sgRNAs. A barcode plot, highlighting the rank of sgRNAs for a given gene relative to the entire data set is generated for the top-ranked gene (19612). Abundance of sgRNAs targeting this gene tend to increase with drug treatment at 0.05 FDR.

```
> genesymbols <- x$genes[,3]
> genesymbollist <- list()
> unq <- unique(genesymbols)
> unq <- unq[!is.na(unq)]
> for(i in unq) {
+   sel <- genesymbols==i & !is.na(genesymbols)
+   if(sum(sel)>3)
+     genesymbollist[[i]] <- which(sel)
+ }
> x$common.dispersion <- fit$dispersion
> camera.res <- camera(x, index=genesymbollist, des, contrast=2)
> camera.res[1:10,]
      NGenes Direction   PValue    FDR
19612      5        Up 9.24e-07 0.0051
8808       4        Up 6.92e-05 0.1607
8370       4        Up 8.75e-05 0.1607
10386      4        Up 2.02e-04 0.2788
4086       4        Up 2.99e-04 0.3294
10784      4        Up 3.89e-04 0.3571
2005       4        Up 6.82e-04 0.4981
11531      4        Up 7.23e-04 0.4981
4635       5        Up 9.56e-04 0.5200
```

```
1874       4        Up 9.95e-04 0.5200
```

We make a barcode plot for an example (Gene 19612) that ranks highly.

```
> barcodeplot(qlf$table$logFC,index=genesymbollist[[19612]],
+             main="Barcodeplot for Gene 19612",
+             labels=c("Negative logFC", "Positive logFC"),
+             quantile=c(-0.5,0.5))
```



The raw data from this example and several other case studies for this technology can be found at https://bioinf.wehi.edu.au/shRNAseq/.

4.7.7 Setup

This analysis was conducted on:

```
> sessionInfo()

R version 4.4.0 (2024-04-24 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 10 x64 (build 19045)

Matrix products: default


locale:
[1] LC_COLLATE=English_Australia.utf8  LC_CTYPE=English_Australia.utf8
[3] LC_MONETARY=English_Australia.utf8 LC_NUMERIC=C
[5] LC_TIME=English_Australia.utf8

time zone: Australia/Sydney
tzcode source: internal

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

other attached packages:
```

```
[1] edgeR_4.1.30     limma_3.59.10    knitr_1.46       BiocStyle_2.31.0

loaded via a namespace (and not attached):
 [1] digest_0.6.35       fastmap_1.1.1       xfun_0.43
 [4] lattice_0.22-6      splines_4.4.0       htmltools_0.5.8.1
 [7] rmarkdown_2.26      cli_3.6.2           grid_4.4.0
[10] statmod_1.5.0       compiler_4.4.0      highr_0.10
[13] tools_4.4.0         evaluate_0.23       Rcpp_1.0.12
[16] yaml_2.3.8          locfit_1.5-9.9      BiocManager_1.30.22
[19] rlang_1.1.3
```

4.7.8 Acknowledgements

Thanks to Dr Sam Wormald from the WEHI for providing the data set used in this case study.

4.8 Bisulfite sequencing of mouse oocytes

4.8.1 Introduction

The bisulfite sequencing (BS-seq) data of this case study is described in Gahurova *et al.* [18]. The sequence and count data are publicly available from the Gene Expression Omnibus (GEO) at the series accession number GSE86297.

This study investigates the onset and progression of *de novo* methylation. Growing oocytes from pre-pubertal mouse ovaries (post-natal days 7-18) isolated and sorted into the following, non-overlapping size categories: 40-45, 50-55 and 60-65μm with two biological replicates in each. Methylation maps were generated by bisulfite conversion of oocyte DNA and Illumina sequencing. Reduced representation bisulfite sequencing (RRBS [32]) was applied for focusing coverage of CGIs and other GC-rich sequences in all three size classes of oocytes. RRBS reads were trimmed to remove poor quality calls and adapters using Trim Galore and mapped to the mouse genome GRCm38 assembly by Bismark [22]. This is summarized in the table below.

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

```
> targets
         GEO    Sample Group                                        File
1 GSM2299710 40-45um-A  40um GSM2299710_RRBS_40-45oocyte_LibA.cov.txt.gz
2 GSM2299711 40-45um-B  40um GSM2299711_RRBS_40-45oocyte_LibB.cov.txt.gz
3 GSM2299712 50-55um-A  50um GSM2299712_RRBS_50-55oocyte_LibA.cov.txt.gz
4 GSM2299713 50-55um-B  50um GSM2299713_RRBS_50-55oocyte_LibB.cov.txt.gz
5 GSM2299714 60-65um-A  60um GSM2299714_RRBS_60-65oocyte_LibA.cov.txt.gz
6 GSM2299715 60-65um-B  60um GSM2299715_RRBS_60-65oocyte_LibB.cov.txt.gz
```

4.8.2 Reading in the data

The Bismark outputs of the data include one coverage file of the methylation in CpG context for each sample. The coverage file for each of the six samples is available for download at GEO. The first six rows of the coverage output for the first sample are shown below.

```
> s1 <- read.delim(file="GSM2299710_RRBS_40-45oocyte_LibA.cov.txt.gz",
+     header=FALSE, nrows=6)
```

```
> s1
  V1      V2      V3   V4 V5   V6
1  6 3121266 3121266 0.00  0   17
2  6 3121296 3121296 0.00  0   17
3  6 3179319 3179319 1.28  1   77
4  6 3180316 3180316 4.55  1   21
5  6 3182928 3182928 4.33 22  486
6  6 3182937 3182937 5.37 61 1074
```

The six columns (from left to right) represent: chromosome, start position, end position, methylation proportion in percentage, number of methylated C's and number of unmethylated C's. Since the start and end positions of a CpG site from Bismark are the same, we can keep only one of them. The last two columns of counts are we will use for the analysis.

We read in the coverage files of all six samples using `readBismark2DGE`. A DGEList object is created using the count table, and the chromosome number and positions are used for annotation.

```
> files <- targets$File
> yall <- readBismark2DGE(files, sample.names=targets$Sample)
```

The edgeRpackage stores the counts and associated annotation in a DGEList object. There is a row for each CpG locus found in any of the files. There are columns of methylated and unmethylated counts for each sample. The chromosomes and genomic loci are stored in the `genes` component.

```
> yall

An object of class "DGEList"
$counts
          40-45um-A-Me 40-45um-A-Un 40-45um-B-Me 40-45um-B-Un 50-55um-A-Me
6-3121266            0           17            0            4            0
6-3121296            0           17            0            4            0
6-3179319            1           77            0           76            2
6-3180316            1           21            0            0            1
6-3182928           22          486            8          953            7
          50-55um-A-Un 50-55um-B-Me 50-55um-B-Un 60-65um-A-Me 60-65um-A-Un
6-3121266           17            0            0            3            3
6-3121296           16            0            0            0            6
6-3179319           52            0            7           10           43
6-3180316            7            0            0            2            4
6-3182928          714           32         1190           10          618
          60-65um-B-Me 60-65um-B-Un
6-3121266            0           11
```

```
6-3121296            0          11
6-3179319            3          30
6-3180316            1           0
6-3182928           12         651
2271667 more rows ...

$samples
             group lib.size norm.factors
40-45um-A-Me     1  1231757            1
40-45um-A-Un     1 36263318            1
40-45um-B-Me     1  1719267            1
40-45um-B-Un     1 55600556            1
50-55um-A-Me     1  2691638            1
7 more rows ...

$genes
          Chr   Locus
6-3121266   6 3121266
6-3121296   6 3121296
6-3179319   6 3179319
6-3180316   6 3180316
6-3182928   6 3182928
2271667 more rows ...
```

```
> dim(yall)
```

```
[1] 2271672      12
```

We remove the mitochondrial genes as they are usually of less interest.

```
> table(yall$genes$Chr)
```

```
     6      9     17      1      3     13     10      2      4      5     11
111377 120649 101606 140819 108466  95196 116980 173357 157628 159979 161754
    18     16      7      8     14     19      X     12     15      Y     MT
 71737  70964 140225 130786  84974  70614  58361  95580  99646    662    312
```

```
> yall <- yall[yall$genes$Chr!="MT", ]
```

For convenience, we sort the DGEList so that all loci are in genomic order, from chromosome 1 to chromosome Y.

```
> ChrNames <- c(1:19,"X","Y")
> yall$genes$Chr <- factor(yall$genes$Chr, levels=ChrNames)
> o <- order(yall$genes$Chr, yall$genes$Locus)
> yall <- yall[o,]
```

We now annotate the CpG loci with the identity of the nearest gene. We search for the gene transcriptional start site (TSS) closest to each our CpGs:

```
> TSS <- nearestTSS(yall$genes$Chr, yall$genes$Locus, species="Mm")
> yall$genes$EntrezID <- TSS$gene_id
> yall$genes$Symbol <- TSS$symbol
```

```
> yall$genes$Strand <- TSS$strand
> yall$genes$Distance <- TSS$distance
> yall$genes$Width <- TSS$width
> head(yall$genes)
```

```
          Chr   Locus EntrezID Symbol Strand Distance  Width
1-3003886   1 3003886   497097   Xkr4      -  -667612 457017
1-3003899   1 3003899   497097   Xkr4      -  -667599 457017
1-3020877   1 3020877   497097   Xkr4      -  -650621 457017
1-3020891   1 3020891   497097   Xkr4      -  -650607 457017
1-3020946   1 3020946   497097   Xkr4      -  -650552 457017
1-3020988   1 3020988   497097   Xkr4      -  -650510 457017
```

Here `EntrezID`, `Symbol`, `Strand` and `Width` are the Entrez Gene ID, symbol, strand and width of the nearest gene. `Distance` is the genomic distance from the CpG to the TSS. Positive values means the TSS is downstream of the CpG and negative values means the TSS is upstream.

4.8.3 Filtering and normalization

We now turn to statistical analysis of differential methylation. Our first analysis will be for individual CpG loci.

CpG loci that have low coverage are removed prior to downstream analysis as they provide little information for assessing methylation levels. We sum up the counts of methylated and unmethylated reads to get the total read coverage at each CpG site for each sample:

```
> Methylation <- gl(2,1,ncol(yall), labels=c("Me","Un"))
> Me <- yall$counts[, Methylation=="Me"]
> Un <- yall$counts[, Methylation=="Un"]
> Coverage <- Me + Un
> head(Coverage)
```

```
          40-45um-A-Me 40-45um-B-Me 50-55um-A-Me 50-55um-B-Me 60-65um-A-Me
1-3003886            0            0            0            0            3
1-3003899            0            0            0            0            3
1-3020877           84           77          114           21           86
1-3020891           84           78          116           21           86
1-3020946          146          369          210          165          195
1-3020988           38           91           60           94           50
          60-65um-B-Me
1-3003886            0
1-3003899            0
1-3020877           57
1-3020891           57
1-3020946          168
1-3020988           25
```

As a conservative rule of thumb, we require a CpG site to have a total count (both methylated and unmethylated) of at least 8 in every sample before it is considered in the study.

```
> HasCoverage <- rowSums(Coverage >= 8) == 6
```

This filtering criterion could be relaxed somewhat in principle but the number of CpGs kept in the analysis is large enough for our purposes.

We also filter out CpGs that are never methylated or always methylated as they provide no information about differential methylation:

```
> HasBoth <- rowSums(Me) > 0 & rowSums(Un) > 0
> table(HasCoverage, HasBoth)

           HasBoth
HasCoverage   FALSE    TRUE
      FALSE 1601772  295891
      TRUE   118785  254912
```

The DGEList object is subsetted to retain only the non-filtered loci:

```
> y <- yall[HasCoverage & HasBoth,, keep.lib.sizes=FALSE]
```

A key difference between BS-seq and other sequencing data is that the pair of libraries holding the methylated and unmethylated reads for a particular sample are treated as a unit. To ensure that the methylated and unmethylated reads for the same sample are treated on the same scale, we need to set the library sizes to be equal for each pair of libraries. We set the library sizes for each sample to be the average of the total read counts for the methylated and unmethylated libraries:

```
> TotalLibSize <- 0.5 * y$samples$lib.size[Methylation=="Me"] +
+                 0.5 * y$samples$lib.size[Methylation=="Un"]
> y$samples$lib.size <- rep(TotalLibSize, each=2)
> y$samples

             group lib.size norm.factors
40-45um-A-Me     1 10427408            1
40-45um-A-Un     1 10427408            1
40-45um-B-Me     1 19792269            1
40-45um-B-Un     1 19792269            1
50-55um-A-Me     1 11322495            1
50-55um-A-Un     1 11322495            1
50-55um-B-Me     1 12632062            1
50-55um-B-Un     1 12632062            1
60-65um-A-Me     1  9487110            1
60-65um-A-Un     1  9487110            1
60-65um-B-Me     1 10231167            1
60-65um-B-Un     1 10231167            1
```

Other normalization methods developed for RNA-seq data are not required for BS-seq data.

4.8.4 Data exploration

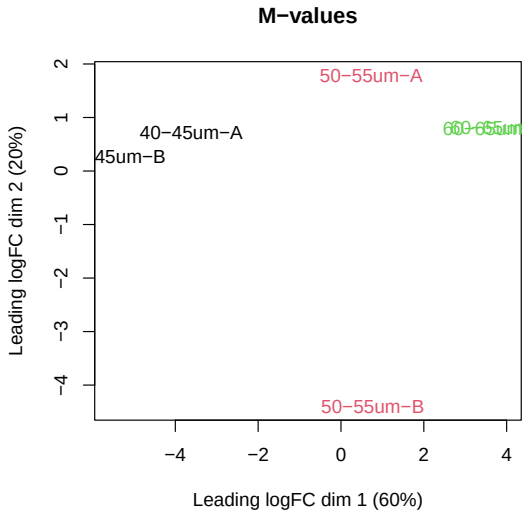
The data can be explored by generating multi-dimensional scaling (MDS) plots on the methylation level (M-value) of the CpG sites. The M-value is calcualted by the log of the ratio of methylated and unmethylated C's, which is equivalent to the difference between methylated and unmethylated C's on the log-scale [13]. A prior count of 2 is added to avoid logarithms of zero.

```
> Me <- y$counts[, Methylation=="Me"]
> Un <- y$counts[, Methylation=="Un"]
> M <- log2(Me + 2) - log2(Un + 2)
> colnames(M) <- targets$Sample
```

Here `M` contains the empirical logit methylation level for each CpG site in each sample. We have used a prior count of 2 to avoid logarithms of zero.

Now we can generate a multi-dimensional scaling (MDS) plot to explore the overall differences between the methylation levels of the different samples.

```
> plotMDS(M, col=rep(1:3, each=2), main="M-values")
```



Replicate samples cluster together within the 40-45 and 60-65μm categories but are far apart in the 50-55μm group. The plot also indicates a huge difference in methylation level between the 40-45 and 60-65μm groups.

4.8.5 Design matrix

One aim of this study is to identify differentially methylated (DM) loci between the different cell populations. In edgeR, this can be done by fitting linear models under a specified design matrix and testing for corresponding coefficients or contrasts. A basic sample-level design matrix can be made as follows:

```
> designSL <- model.matrix(~0+Group, data=targets)
> designSL

  Group40um Group50um Group60um
1         1         0         0
2         1         0         0
3         0         1         0
4         0         1         0
5         0         0         1
6         0         0         1
```

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

The we expand this to the full design matrix modeling the sample and methylation effects:

```
> design <- modelMatrixMeth(designSL)
> design
   Sample1 Sample2 Sample3 Sample4 Sample5 Sample6 Group40um Group50um
1        1       0       0       0       0       0         1         0
2        1       0       0       0       0       0         0         0
3        0       1       0       0       0       0         1         0
4        0       1       0       0       0       0         0         0
5        0       0       1       0       0       0         0         1
6        0       0       1       0       0       0         0         0
7        0       0       0       1       0       0         0         1
8        0       0       0       1       0       0         0         0
9        0       0       0       0       1       0         0         0
10       0       0       0       0       1       0         0         0
11       0       0       0       0       0       1         0         0
12       0       0       0       0       0       1         0         0
   Group60um
1          0
2          0
3          0
4          0
5          0
6          0
7          0
8          0
9          1
10         0
11         1
12         0
```

The first six columns represent the sample coverage effects. The last three columns represent the methylation levels (in logit units) in the three groups.

4.8.6 Differential methylation analysis at CpG loci

For simplicity, we only consider the CpG methylation in chromosome 1. We subset the coverage files so that they only contain methylation information of the first chromosome.

```
> y1 <- y[y$genes$Chr==1, ]
```

Then we proceed to testing for differentially methylated CpG sites between different groups. We fit quasi NB GLM for all the CpG loci using the `glmQLFit` function.

```
> fit <- glmQLFit(y1, design)
```

We identify differentially methylated CpG loci between the 40-45 and 60-65μm group using the likelihood-ratio test. The contrast corresponding to this comparison is constructed using the `makeContrasts` function.

```
> contr <- makeContrasts(
+     Group60vs40 = Group60um - Group40um, levels=design)
> qlf <- glmQLFTest(fit, contrast=contr)
```

The top set of most significant DMRs can be examined with `topTags`. Here, positive log-fold changes represent CpG sites that have higher methylation level in the 60-65μm group compared to the 40-45μm group. Multiplicity correction is performed by applying the Benjamini-Hochberg method on the p-values, to control the false discovery rate (FDR).

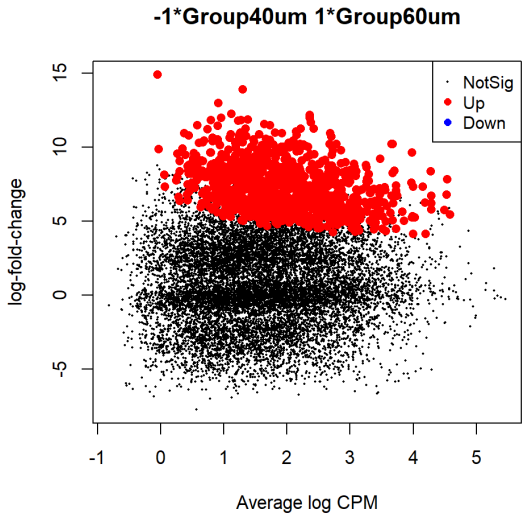
```
> topTags(qlf)

Coefficient:  -1*Group40um 1*Group60um
            Chr     Locus  EntrezID        Symbol Strand Distance Width logFC
1-120170060   1 120170060     73103 3110009E18Rik      +   -48873 67003  7.86
1-183357406   1 183357406    338366          Mia3      -   -12159 43330  8.40
1-131987595   1 131987595    212980       Slc45a3      +   -16986 12364 10.74
1-120169950   1 120169950     73103 3110009E18Rik      +   -48763 67003  9.23
1-172206570   1 172206570     18611        Pea15a      -     -234 10077 10.10
1-183357373   1 183357373    338366          Mia3      -   -12192 43330  7.35
1-169954561   1 169954561     15490       Hsd17b7      -   -14644 19669 12.22
1-92943747    1  92943747 100037260 9430060I03Rik      -    -6843 17048  9.57
1-153126749   1 153126749     16782         Lamc2      -   -59698 63692  8.22
1-36500213    1  36500213     94218         Cnnm3      +    11654 16370  9.95
            logCPM    F   PValue      FDR
1-120170060   4.54 40.2 2.28e-10 3.25e-06
1-183357406   4.28 38.3 6.07e-10 4.32e-06
1-131987595   2.70 33.5 7.33e-09 3.48e-05
1-120169950   3.47 32.3 1.31e-08 4.53e-05
1-172206570   2.71 31.7 1.84e-08 4.53e-05
1-183357373   4.15 31.6 1.91e-08 4.53e-05
1-169954561   2.36 28.9 7.63e-08 1.39e-04
1-92943747    2.36 28.7 8.66e-08 1.39e-04
1-153126749   3.29 28.5 9.41e-08 1.39e-04
1-36500213    2.20 28.3 1.06e-07 1.39e-04
```

The total number of DMRs in each direction at a FDR of 5% can be examined with `decideTests`.

```
> summary(decideTests(qlf))

       -1*Group40um 1*Group60um
Down                          0
NotSig                    13242
Up                          993
```

The differential methylation results can be visualized using an MD plot. The difference of the M-value for each CpG site is plotted against the average abundance of that CpG site. Significantly DMRs at a FDR of 5% are highlighted.

```
> plotMD(qlf)
```



It can be seen that most of the DMRs have higher methylation levels in 60-65μm group compared to the 40-45μm group. This is consistent with the findings in Gahurova *et al.* [18].

4.8.7 Summarizing counts in promoter regions

It is usually of great biological interest to examine the methylation level within the gene promoter regions. For simplicity, we define the promoter of a gene as the region from 2kb upstream to 1kb downstream of the transcription start site of that gene. We then subset the CpGs to those contained in a promoter region.

```
> InPromoter <- yall$genes$Distance >= -1000 & yall$genes$Distance <= 2000
> yIP <- yall[InPromoter,,keep.lib.sizes=FALSE]
```

We compute the total counts for each gene promoter:

```
> ypr <- rowsum(yIP, yIP$genes$EntrezID, reorder=FALSE)
> ypr$genes$EntrezID <- NULL
```

The integer matrix `ypr$counts` contains the total numbers of methylated and unmethylated CpGs observed within the promoter of each gene.

Filtering is performed in the same way as before. We sum up the read counts of both methylated and unmethylated Cs at each gene promoter within each sample.

```
> Mepr <- ypr$counts[,Methylation=="Me"]
> Unpr <- ypr$counts[,Methylation=="Un"]
> Coveragepr <- Mepr + Unpr
```

Since each row represents a 3,000-bps-wide promoter region that contains multiple CpG sites, we would expect less filtering than before.

```
> HasCoveragepr <- rowSums(Coveragepr >= 8) == 6
> HasBothpr <- rowSums(Mepr) > 0 & rowSums(Unpr) > 0
> table(HasCoveragepr, HasBothpr)

             HasBothpr
HasCoveragepr FALSE  TRUE
        FALSE  3656  3053
        TRUE     85 15043

> ypr <- ypr[HasCoveragepr & HasBothpr,,keep.lib.sizes=FALSE]
```

Same as before, we do not perform normalization but set the library sizes for each sample to be the average of the total read counts for the methylated and unmethylated libraries.

```
> TotalLibSizepr <- 0.5 * ypr$samples$lib.size[Methylation=="Me"] +
+                   0.5 * ypr$samples$lib.size[Methylation=="Un"]
> ypr$samples$lib.size <- rep(TotalLibSizepr, each=2)
> ypr$samples

             group lib.size norm.factors
40-45um-A-Me     1  8015762            1
40-45um-A-Un     1  8015762            1
40-45um-B-Me     1 11768442            1
40-45um-B-Un     1 11768442            1
50-55um-A-Me     1  9989109            1
50-55um-A-Un     1  9989109            1
50-55um-B-Me     1  8506420            1
50-55um-B-Un     1  8506420            1
60-65um-A-Me     1  8089503            1
60-65um-A-Un     1  8089503            1
60-65um-B-Me     1  6500102            1
60-65um-B-Un     1  6500102            1
```

4.8.8 Differential methylation in gene promoters

We fit quasi NB GLMs for all the gene promoters using `glmQLFit`.

```
> fitpr <- glmQLFit(ypr, design)
```

Then we can proceed to testing for differential methylation in gene promoter regions between different populations. Suppose the comparison of interest is the same as before. The same contrast can be used for the testing.

```
> qlfpr <- glmQLFTest(fitpr, contrast=contr)
```

The top set of most differentially methylated gene promoters can be viewed with `topTags`:

```
> topTags(qlfpr, n=20)

Coefficient:  -1*Group40um 1*Group60um
          Chr        Symbol Strand logFC logCPM     F   PValue      FDR
```

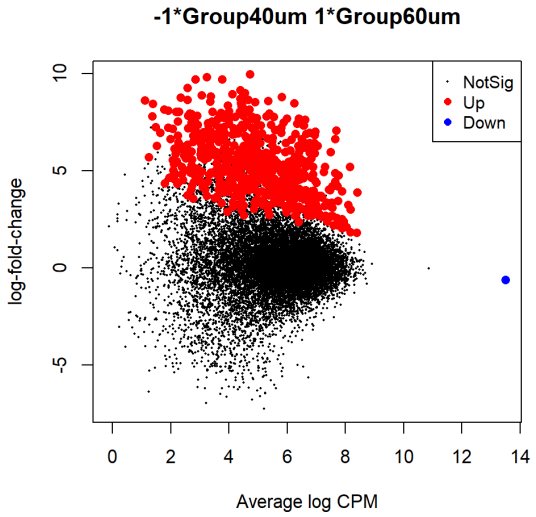
```
237336     10         Tbpl1      -  7.07   7.69 126.9 5.43e-12 4.78e-08
30841       5         Kdm2b      -  6.64   7.64 125.2 6.35e-12 4.78e-08
109934     11           Abr      -  5.20   8.15 105.1 4.71e-11 2.30e-07
15552       4         Htr1d      +  7.02   6.96 102.3 6.40e-11 2.30e-07
210274      7        Shank2      +  7.32   6.56 100.7 7.65e-11 2.30e-07
78102      15 8430426J06Rik      -  7.78   5.53  93.3 1.78e-10 4.05e-07
246257     11         Ovca2      -  7.62   6.60  91.6 2.11e-10 4.05e-07
69727       5         Usp46      -  5.95   7.49  91.5 2.21e-10 4.05e-07
20410      14        Sorbs3      -  6.43   6.92  90.7 2.42e-10 4.05e-07
101100      6         Ttll3      +  7.70   6.40  89.3 2.90e-10 4.36e-07
217198     11       Plekhh3      -  6.95   6.76  86.8 3.92e-10 5.24e-07
212307     18        Mapre2      +  6.36   6.96  86.3 4.18e-10 5.24e-07
72446       2         Prr5l      -  7.34   6.54  82.3 6.73e-10 7.79e-07
83397      10        Akap12      +  6.44   6.76  81.7 7.48e-10 8.04e-07
102465670  11       Mir7115      +  8.18   4.51  80.9 8.33e-10 8.35e-07
114483644  17      Mir3083b      +  8.80   5.81  76.9 9.17e-10 8.62e-07
18611       1        Pea15a      -  7.21   5.79  78.8 1.10e-09 9.37e-07
16905       3          Lmna      -  6.02   6.79  78.4 1.16e-09 9.37e-07
54397      17          Ppt2      -  5.97   6.86  77.8 1.25e-09 9.37e-07
100038353  18       Gm10532      +  7.76   4.79  77.5 1.28e-09 9.37e-07
```

The total number of DM gene promoters identified at an FDR of 5% can be shown with `decideTests`.

```
> summary(decideTests(qlfpr))

       -1*Group40um 1*Group60um
Down                          1
NotSig                    14426
Up                          616
```

The differential methylation results can be visualized with an MD plot.

```
> plotMD(qlfpr)
```



4.8.9 Setup

This analysis was conducted on:

```
> sessionInfo()

R version 4.4.0 (2024-04-24 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 10 x64 (build 19045)

Matrix products: default


locale:
[1] LC_COLLATE=English_Australia.utf8  LC_CTYPE=English_Australia.utf8
[3] LC_MONETARY=English_Australia.utf8 LC_NUMERIC=C
[5] LC_TIME=English_Australia.utf8

time zone: Australia/Sydney
tzcode source: internal

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

other attached packages:
[1] knitr_1.46       BiocStyle_2.31.0 edgeR_4.1.28     limma_3.59.10

loaded via a namespace (and not attached):
 [1] utf8_1.2.4              RSQLite_2.3.6           lattice_0.22-6
 [4] hms_1.1.3               digest_0.6.35           magrittr_2.0.3
 [7] evaluate_0.23           grid_4.4.0              fastmap_1.1.1
[10] blob_1.2.4              jsonlite_1.8.8          AnnotationDbi_1.65.2
[13] GenomeInfoDb_1.39.14    DBI_1.2.2               BiocManager_1.30.22
[16] httr_1.4.7              fansi_1.0.6             UCSC.utils_0.99.7
[19] Biostrings_2.71.6       cli_3.6.2               rlang_1.1.3
[22] crayon_1.5.2            XVector_0.43.1          Biobase_2.63.1
[25] splines_4.4.0           bit64_4.0.5             cachem_1.0.8
[28] yaml_2.3.8              tools_4.4.0             parallel_4.4.0
[31] tzdb_0.4.0              memoise_2.0.1           GenomeInfoDbData_1.2.12
[34] locfit_1.5-9.9          BiocGenerics_0.49.1     vctrs_0.6.5
[37] R6_2.5.1                png_0.1-8               stats4_4.4.0
[40] lifecycle_1.0.4         zlibbioc_1.49.3         KEGGREST_1.43.0
[43] S4Vectors_0.41.7        IRanges_2.37.1          bit_4.0.5
[46] vroom_1.6.5             pkgconfig_2.0.3         pillar_1.9.0
[49] glue_1.7.0              Rcpp_1.0.12             statmod_1.5.0
[52] xfun_0.43               tibble_3.2.1            tidyselect_1.2.1
[55] highr_0.10              org.Mm.eg.db_3.19.1     htmltools_0.5.8.1
[58] rmarkdown_2.26          readr_2.1.5             compiler_4.4.0
```

4.9 Time course RNA-seq experiments of Drosophila melanogaster

4.9.1 Introduction

The data for this case study was generated by Graveley *et al.* [19] and was previously analyzed by Law *et al.* [23] using polynomial regression. Here we reanalyze the data using smoothing splines to illustrate a general approach that can be taken to time-course data with many time points. The approach taken here does not require biological replicates at each time point — we can instead estimate the magnitude of biological variation from the smoothness or otherwise of the time-course expression trend for each gene.

Graveley *et al.* conducted RNA-seq to examine the dynamics of gene expression throughout developmental stages of the common fruit fly (*Drosophila melanogaster*). 30 whole-animal samples representing 27 distinct stages of development were used for sequencing. These included 12 embryonic samples collected at 2-hour intervals from 0–2 hours to 22–24 hours and also six larval, six pupal and three sexed adult stages at 1, 5 and 30 days after eclosion. Each biological sample was sequenced several times and we view these as technical replicates. Here we analyze only the data from the 12 embryonic stages.

RNA-seq read counts for this data are available from the ReCount [15] at . http://bowtie-bio.sourceforge.net. The table of read counts can be read into R directly from the ReCount website by

```
> CountFile <- paste("http://bowtie-bio.sourceforge.net/recount",
+                    "countTables",
+                    "modencodefly_count_table.txt", sep="/")
> Counts <- read.delim(CountFile, row.names=1)
```

The sample information can be read by

```
> SampleFile <- paste("http://bowtie-bio.sourceforge.net/recount",
+                     "phenotypeTables",
+                     "modencodefly_phenodata.txt", sep="/")
> Samples <- read.delim(SampleFile, row.names=1, sep=" ", stringsAsFactors=FALSE)
```

The data has 127 columns of counts but only 30 biologically independent samples:

```
> dim(Counts)

[1] 14869   147
```

We therefore sum the technical replicates to get total genewise read counts for each biological sample:

```
> PooledCounts <- sumTechReps(Counts, ID=Samples$stage)
> dim(PooledCounts)

[1] 14869    30

> colnames(PooledCounts)

 [1] "Embryos0002"          "Embryos0204"          "Embryos0406"
 [4] "Embryos0608"          "Embryos0810"          "Embryos1012"
```

```
 [7] "Embryos1214"          "Embryos1416"          "Embryos1618"
[10] "Embryos1820"          "Embryos2022"          "Embryos2224"
[13] "L1Larvae"             "L2Larvae"             "L3Larvae12hrpostmolt"
[16] "L3LarvaePS12"         "L3LarvaePS36"         "L3LarvaePS79"
[19] "WPP12hr"              "WPP24hr"              "adultfemale1d"
[22] "adultfemale30d"       "adultfemale5d"        "adultmale1d"
[25] "adultmale30d"         "adultmale5d"          "pupaeWPP2d"
[28] "pupaeWPP3d"           "pupaeWPP4d"           "whiteprepupae"
```

4.9.2 DEGList object

The embryonic stages correspond to the first 12 columns of data. We create a `DGEList` object from these columns by:

```
> Hours <- seq(from=2, to=24, by=2)
> Time <- paste0(Hours,"hrs")
> y <- DGEList(counts=PooledCounts[,1:12], group=Time)
```

4.9.3 Gene annotation

We use the *D. melanogaster* organism package `org.Dm.eg.db` to map the flybase gene IDs to gene symbols. Some flybase IDs map to multiple genes. To handle this possibility, we use the multiVals argument of `mapIds` to keep the multiple symbols separated by semicolons:

```
> library(org.Dm.eg.db)
> multiVals <- function(x) paste(x,collapse=";")
> Symbol <- mapIds(org.Dm.eg.db, keys=rownames(y), keytype="FLYBASE",
+                  column="SYMBOL", multiVals=multiVals)
> y$genes <- data.frame(Symbol=Symbol, stringsAsFactors=FALSE)
> head(y$genes)
```

```
                    Symbol
FBgn0000003 7SLRNA:CR32864
FBgn0000008              a
FBgn0000014          abd-A
FBgn0000015          Abd-B
FBgn0000017            Abl
FBgn0000018            abo
```

We will remove flybase IDs that cannot be mapped to gene symbols:

```
> HasSymbol <- y$genes$Symbol != "NA"
> y <- y[HasSymbol, , keep.lib.sizes=FALSE]
```

4.9.4 Filtering and normalization

We filter out lowly expressed genes.

```
> keep <- filterByExpr(y)
> table(keep)
```

```
keep
FALSE  TRUE
 2562 10964
> y <- y[keep, , keep.lib.sizes=FALSE]
```

TMM normalization is performed to estimate effective library sizes:

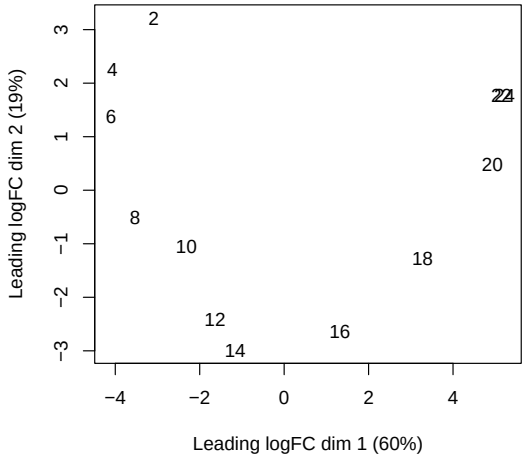
```
> y <- normLibSizes(y)
> y$samples
            group lib.size norm.factors
Embryos0002  2hrs 48258253        0.927
Embryos0204  4hrs 38828271        0.899
Embryos0406  6hrs 89803894        0.942
Embryos0608  8hrs 55475614        1.018
Embryos0810 10hrs 52438317        1.115
Embryos1012 12hrs 67362292        1.124
Embryos1214 14hrs 82936353        1.174
Embryos1416 16hrs 66518905        0.985
Embryos1618 18hrs 61271846        0.968
Embryos1820 20hrs 62003653        0.922
Embryos2022 22hrs 33933605        0.972
Embryos2224 24hrs 68279305        0.997
```

4.9.5 Data exploration

The data can be explored by generating multi-dimensional scaling (MDS) plots. This visualizes the differences between the expression profiles of different samples in two dimensions.

The 12 successive embryonic developmental stages are labelled according to number of hours since fertilization. The MDS plot shows a smooth trend of transition in gene expression during embryonic development from the start up to 22 hours:

```
> plotMDS(y, labels=Hours)
```



4.9.6 Design matrix

The aim of a time course experiment is to examine the relationship between gene abundances and time points. Assuming gene expression changes smoothly over time, we can use a polynomial or a cubic spline curve with a certain number of degrees of freedom to model gene expression along time.

To use a polynomial with, say 3 degrees of freedom, a design matrix could be constructed as follows.

```
> X <- poly(Hours, degree=3)
> design <- model.matrix(~ X)
> design

   (Intercept)      X1       X2      X3
1            1 -0.4599  0.50183 -0.4599
2            1 -0.3763  0.22810  0.0418
3            1 -0.2927  0.00912  0.2927
4            1 -0.2091 -0.15511  0.3484
5            1 -0.1254 -0.26460  0.2648
6            1 -0.0418 -0.31935  0.0976
7            1  0.0418 -0.31935 -0.0976
8            1  0.1254 -0.26460 -0.2648
9            1  0.2091 -0.15511 -0.3484
10           1  0.2927  0.00912 -0.2927
11           1  0.3763  0.22810 -0.0418
12           1  0.4599  0.50183  0.4599
attr(,"assign")
[1] 0 1 1 1
```

Then the coefficients X1, X2 and X3 represent the linear, quadratic and cubic time effect, respectively. Hypotheses can be tested for each one of the coefficients.

To use a cubic regression spline curve with, say 3 degrees of freedom, a design matrix can be constructed as follows.

```
> library(splines)
> X <- ns(Hours, df=3)
> design <- model.matrix(~ X)
> design

   (Intercept)      X1    X2     X3
1            1  0.0000 0.000  0.000
2            1 -0.0643 0.203 -0.135
3            1 -0.0995 0.380 -0.253
4            1 -0.0764 0.503 -0.335
5            1  0.0334 0.547 -0.365
6            1  0.2199 0.512 -0.333
7            1  0.4134 0.435 -0.247
8            1  0.5391 0.359 -0.114
9            1  0.5281 0.323  0.058
10           1  0.3806 0.332  0.260
11           1  0.1417 0.373  0.482
12           1 -0.1429 0.429  0.714
attr(,"assign")
[1] 0 1 1 1
```

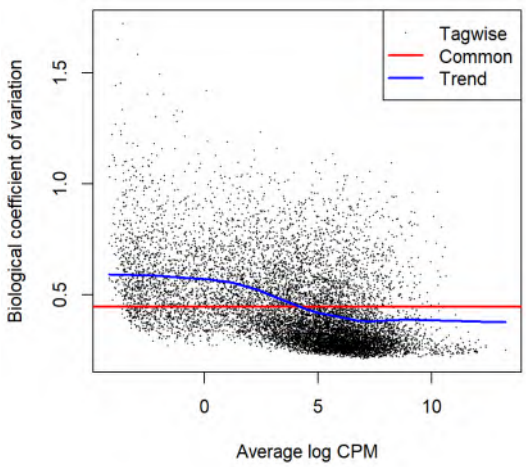
Here the three coefficients do not have any particular meaning. Hypothesis testing would only make sense if the three coefficients are assessed together. The advantage of using a cubic spline curve is that it provides more stable fit at the end points compared to a polynomial.

The spline curve with 3 degrees of freedonm has 2 knots where cubic polynomials are splined together. In general, choosing a number of degrees of freedom to be in range of 3-5 is reasonable. Setting the degrees of freedom equal to 1 would be equivalent to simple linear regression, i.e., a straight line trend.

4.9.7 Dispersion estimation

The NB dispersion is estimated using the `estimateDisp` function. This returns the `DGEList` object with additional entries for the estimated NB dispersion for each gene. These estimates can be visualized with `plotBCV`, which shows the root-estimate, i.e., the biological coefficient of variation for each gene.

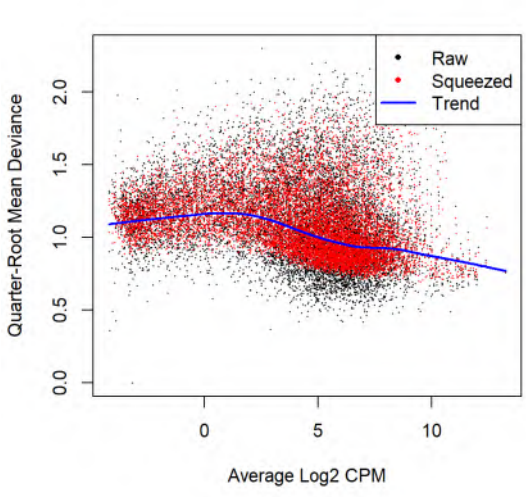
```
> y <- estimateDisp(y, design)
> sqrt(y$common.dispersion)

[1] 0.447

> plotBCV(y)
```



Note that this step is now optional as all the NB dispersion estimates will not be used further under the latest quasi-likelihood (QL) pipeline.

For the QL dispersions, estimation can be performed using the `glmQLFit` function. This returns a `DGEGLM` object containing the estimated values of the GLM coefficients for each gene, as well as the fitted mean-QL dispersion trend, the squeezed QL estimates and the prior degrees of freedom (df). These can be visualized with the `plotQLDisp` function.

```
> fit <- glmQLFit(y, design, robust=TRUE)
> plotQLDisp(fit)
```



4.9.8 Time course trend analysis

In a time course experiment, we are looking for genes that change expression level over time. Here, the design matrix uses 3 natural spline basis vectors to model smooth changes over time, without assuming any particular pattern to the trend. We test for a trend by conducting F-tests on 3 df for each gene:

```
> fit <- glmQLFTest(fit, coef=2:4)
```

The `topTags` function lists the top set of genes with most significant time effects.

```
> tab <- as.data.frame(topTags(fit, n=30))
> tab
```

	Symbol	logFC.X1	logFC.X2	logFC.X3	logCPM	F	PValue	FDR
FBgn0031930	CG7025	7.5090	8.16	12.45	3.31	414	4.06e-16	4.46e-12
FBgn0036851	CG14082	10.2027	4.98	9.01	1.53	195	2.14e-13	8.43e-10
FBgn0032024	CG14273	9.4952	7.31	10.28	2.67	245	2.45e-13	8.43e-10
FBgn0053543	CG33543	9.6714	5.66	10.82	2.91	215	3.08e-13	8.43e-10
FBgn0259896	NimC1	9.7137	12.35	9.98	3.89	328	4.91e-13	1.08e-09
FBgn0030157	CG1468	6.8802	6.77	13.21	4.00	172	1.06e-12	1.93e-09
FBgn0023179	amon	9.0988	8.65	9.82	5.00	493	1.61e-12	2.53e-09
FBgn0004169	up	9.8544	8.81	8.76	10.53	476	1.97e-12	2.64e-09
FBgn0030747	CG4301	8.9327	10.52	8.54	6.05	474	2.17e-12	2.64e-09
FBgn0038606	CG15803	10.1116	5.42	9.88	2.76	189	3.03e-12	3.23e-09
FBgn0002773	Mlc2	9.6207	9.05	8.56	11.32	424	3.88e-12	3.23e-09
FBgn0001114	Glt	9.3347	8.16	8.25	9.48	420	4.08e-12	3.23e-09
FBgn0027556	CG4928	7.9255	9.77	10.92	7.42	418	4.35e-12	3.23e-09
FBgn0025833	CG8910	9.7922	8.85	10.27	2.70	170	4.36e-12	3.23e-09
FBgn0051265	CG31265	7.1627	2.92	13.02	3.43	133	4.42e-12	3.23e-09
FBgn0035293	CG5687	8.2325	8.81	10.45	4.99	400	4.93e-12	3.24e-09
FBgn0003086	Pig1	0.0567	7.17	11.39	1.27	103	5.02e-12	3.24e-09
FBgn0051664	CG31664	8.3922	4.17	11.28	2.54	127	5.57e-12	3.39e-09
FBgn0039325	CG10560	3.1161	7.04	12.80	2.93	118	6.81e-12	3.93e-09
FBgn0028699	Rh50	5.6852	3.86	13.26	3.60	119	9.34e-12	4.88e-09
FBgn0033958	jef	9.6893	7.72	9.48	5.69	370	9.35e-12	4.88e-09
FBgn0004516	Gad1	8.4619	8.74	9.72	6.19	364	9.95e-12	4.96e-09
FBgn0030928	CG15044	8.6350	11.91	11.24	3.61	175	1.27e-11	5.97e-09
FBgn0038420	CG10311	8.1302	7.46	9.57	5.82	346	1.33e-11	5.97e-09
FBgn0033250	CG14762	8.8984	9.75	8.92	5.68	342	1.44e-11	5.97e-09
FBgn0030897	Frq1	8.8878	7.33	10.31	3.50	230	1.44e-11	5.97e-09
FBgn0053179	beat-IIIb	8.1448	9.78	9.42	2.70	192	1.55e-11	5.97e-09
FBgn0010482	hlk	9.2026	9.37	10.28	5.30	328	1.60e-11	5.97e-09
FBgn0250908	beat-VII	8.6855	12.76	9.57	2.83	188	1.62e-11	5.97e-09
FBgn0030543	CG11585	4.1094	4.34	13.29	3.08	106	1.66e-11	5.97e-09

The total number of genes with significant (5% FDR) changes at different time points can be examined with `decideTests`.

```
> summary(decideTests(fit))
```

	X3-X2-X1
NotSig	1708
Sig	9256

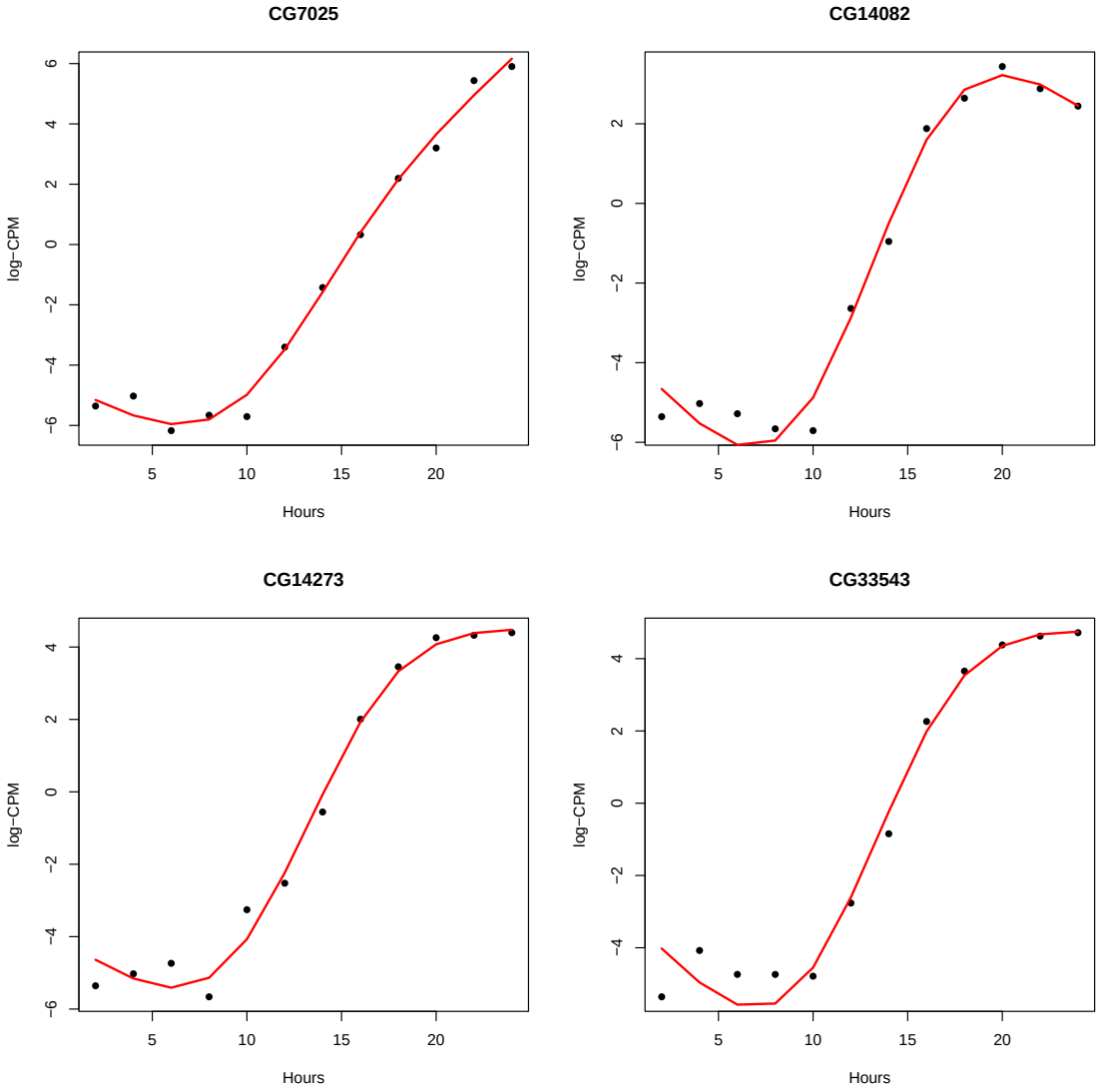
Note that all three spline coefficients should be tested together in this way. It is not meaningful to replace the F-tests with t-tests for the individual coefficients, and similarly the `logFC` columns of the top table do not have any interpretable meaning. The trends should instead be interpreted by way of trend plots, as we show now.

Finally, we visualize the fitted spline curves for the top four genes. We start by computing the observed and fitted log-CPM values for each gene:

```
> logCPM.obs <- cpm(y, log=TRUE, prior.count=fit$prior.count)
> logCPM.fit <- cpm(fit, log=TRUE)
```

We then loop through the first four genes in the topTags table, plotting the observed and fitted values for each gene:

```
> par(mfrow=c(2,2))
> for(i in 1:4) {
+   FlybaseID <- row.names(tab)[i]
+   Symbol <- tab$Symbol[i]
+   logCPM.obs.i <- logCPM.obs[FlybaseID,]
+   logCPM.fit.i <- logCPM.fit[FlybaseID,]
+   plot(Hours, logCPM.obs.i, ylab="log-CPM", main=Symbol, pch=16)
+   lines(Hours, logCPM.fit.i, col="red", lwd=2)
+ }
```




```
> par(mfrow=c(1,1))
```

In each plot, the red curve shows the fitted log2-CPM for that genes while the black dots show the observed log-CPM values. All four of the genes show increased expression during embryo development, with an up trend especially during the period from 6hrs to 20hrs.

4.9.9 Setup

This analysis was conducted using the following software setup:

```
> sessionInfo()

R version 4.4.0 (2024-04-24 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 10 x64 (build 19045)
```

```
Matrix products: default


locale:
[1] LC_COLLATE=English_Australia.utf8  LC_CTYPE=English_Australia.utf8
[3] LC_MONETARY=English_Australia.utf8 LC_NUMERIC=C
[5] LC_TIME=English_Australia.utf8

time zone: Australia/Sydney
tzcode source: internal

attached base packages:
[1] splines   stats4    stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
 [1] org.Dm.eg.db_3.19.1  AnnotationDbi_1.65.2 IRanges_2.37.1
 [4] S4Vectors_0.41.7     Biobase_2.63.1       BiocGenerics_0.49.1
 [7] edgeR_4.1.28         limma_3.59.10        knitr_1.46
[10] BiocStyle_2.31.0

loaded via a namespace (and not attached):
 [1] bit_4.0.5               jsonlite_1.8.8          compiler_4.4.0
 [4] BiocManager_1.30.22     highr_0.10              crayon_1.5.2
 [7] Rcpp_1.0.12             blob_1.2.4              Biostrings_2.71.6
[10] png_0.1-8               yaml_2.3.8              fastmap_1.1.1
[13] statmod_1.5.0           lattice_0.22-6          R6_2.5.1
[16] XVector_0.43.1          GenomeInfoDb_1.39.14    GenomeInfoDbData_1.2.12
[19] DBI_1.2.2               rlang_1.1.3             KEGGREST_1.43.0
[22] cachem_1.0.8            xfun_0.43               bit64_4.0.5
[25] RSQLite_2.3.6           memoise_2.0.1           cli_3.6.2
[28] zlibbioc_1.49.3         digest_0.6.35           grid_4.4.0
[31] locfit_1.5-9.9          vctrs_0.6.5             evaluate_0.23
[34] rmarkdown_2.26          httr_1.4.7              pkgconfig_2.0.3
[37] UCSC.utils_0.99.7       tools_4.4.0             htmltools_0.5.8.1
```

4.10 Single cell RNA-seq differential expression with pseudo-bulking

4.10.1 Introduction

The single cell RNA-seq data for this case study is from the human breast single cell RNA atlas generated by Pal *et al.* [33]. The preprocessing of the data and the complete bioinformatics analyses of the entire atlas study are described in detail in Chen *et al.* [5]. Most part of the single cell analysis, such as dimensionality reduction and integration, were performed in Seurat [46]. All the generated Seurat objects are publicly available on figshare [7]. Here, we focus on the breast tissue micro-environment from 13 individual healthy donors. The original integrated Seurat object for these 13 healthy donors are available at https://figshare.com/

ndownloader/files/31545890. For demonstration purposes, we created a subset version of the integrated Seurat object that contains 10,000 cells of the total 24,751 cells from the original object.

We first download this subset Seurat object from https://bioinf.wehi.edu.au/edgeR/UserGuideData/SeuratObj.rds.

```
> download.file("https://bioinf.wehi.edu.au/edgeR/UserGuideData/SeuratObj.rds",
+               "SeuratObj.rds")
```

We read in the downloaded Seurat object.

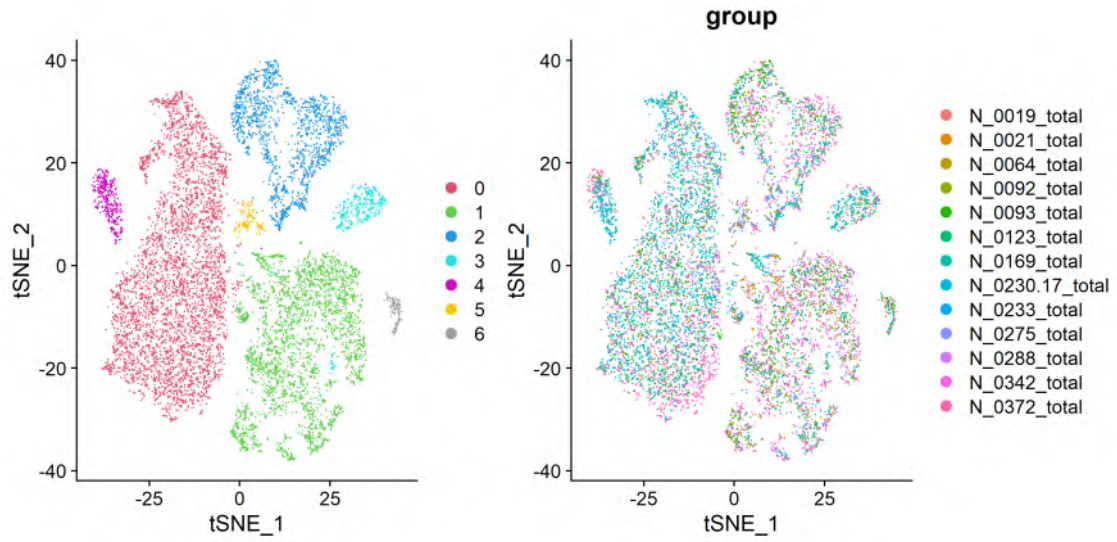
```
> so <- readRDS("SeuratObj.rds")
```

This subset Seurat object contains single cell RNA-seq profiles of 10,000 cells from 13 individual healthy donors. The cell information is stored in the `meta.data` component of the object.

```
> head(so@meta.data)
                                orig.ident nCount_RNA nFeature_RNA        group
N_0019_total_AAACCTGAGGGCTCTC-1          N       7886         2419 N_0019_total
N_0019_total_AAACCTGGTACCGCTG-1          N       2306         1018 N_0019_total
N_0019_total_AAACCTGTCTAGCACA-1          N       8569         2411 N_0019_total
N_0019_total_AAACGGGAGGACAGAA-1          N       3774         1311 N_0019_total
N_0019_total_AAACGGGCATATGAGA-1          N       5689         1753 N_0019_total
N_0019_total_AAAGCAAGTGTAAGTA-1          N       3684         1350 N_0019_total
                                integrated_snn_res.0.05 seurat_clusters
N_0019_total_AAACCTGAGGGCTCTC-1                       1               1
N_0019_total_AAACCTGGTACCGCTG-1                       0               0
N_0019_total_AAACCTGTCTAGCACA-1                       0               0
N_0019_total_AAACGGGAGGACAGAA-1                       0               0
N_0019_total_AAACGGGCATATGAGA-1                       1               1
N_0019_total_AAAGCAAGTGTAAGTA-1                       3               3
```

The t-SNE visualization of the integrated data is shown below. Cells are coloured by cluster on the left and by donor on the right.

```
> p1 <- Seurat::DimPlot(so, reduction="tsne", cols=2:8)
> p2 <- Seurat::DimPlot(so, reduction="tsne", group.by="group")
> p1 | p2
```



4.10.2 Create pseudo-bulk samples

For differential expression analysis of a multi-sample single-cell experiment, pseudo-bulk methods outperform DE methods specifically designed at the single-cell level in terms of both stability and computational speed [9]. Also, the biological variation between samples is preserved and can be assessed in the standard `edgeR` pipeline.

Pseudo-bulk samples are created by aggregating read counts together for all the cells with the same combination of human donor and cluster. Here, we generate pseudo-bulk expression profiles from the Seurat object using the `Seurat2PB` function. The human donor and cell cluster information of the integrated single cell data is stored in the `group` and `seurat_clusters` columns of the `meta.data` component of the Seurat object.

```
> y <- Seurat2PB(so, sample="group", cluster="seurat_clusters")
> dim(y)

[1] 13527    85

> head(y$samples, n=10L)

                      group lib.size norm.factors       sample cluster
N_0019_total_cluster0     1  1679441            1 N_0019_total       0
N_0019_total_cluster1     1  2225898            1 N_0019_total       1
N_0019_total_cluster2     1   350241            1 N_0019_total       2
N_0019_total_cluster3     1   133909            1 N_0019_total       3
N_0019_total_cluster4     1    49889            1 N_0019_total       4
N_0019_total_cluster5     1   160445            1 N_0019_total       5
N_0019_total_cluster6     1    58839            1 N_0019_total       6
N_0021_total_cluster0     1    38263            1 N_0021_total       0
N_0021_total_cluster1     1   515730            1 N_0021_total       1
N_0021_total_cluster2     1    92472            1 N_0021_total       2
```

4.10.3 Filtering and normalization

We first examine the library sizes of all the pseudo-bulk samples and filter out those below the threshold of 50,000.

```
> summary(y$samples$lib.size)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1352   42181  165537  651543  776854 5011510

> keep.samples <- y$samples$lib.size > 5e4
> table(keep.samples)

keep.samples
FALSE  TRUE
   26    59

> y <- y[, keep.samples]
```

We then filter out lowly expressed genes.

```
> keep.genes <- filterByExpr(y, group=y$samples$cluster)
> table(keep.genes)

keep.genes
FALSE  TRUE
 5660  7867

> y <- y[keep.genes, , keep=FALSE]
```

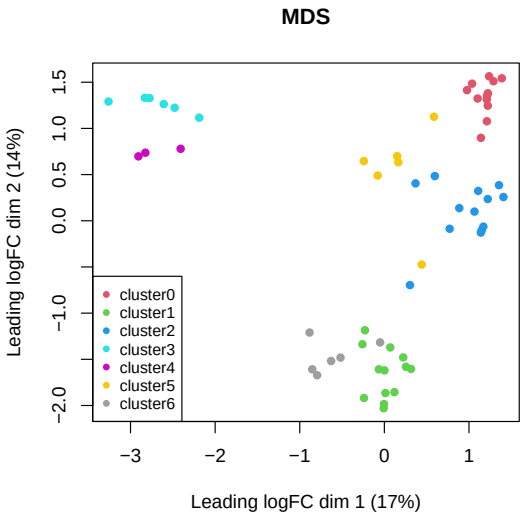
TMM normalization is performed to estimate effective library sizes.

```
> y <- normLibSizes(y)
> head(y$samples, n=10L)

                      group lib.size norm.factors       sample cluster
N_0019_total_cluster0     1  1648290        1.013 N_0019_total       0
N_0019_total_cluster1     1  2183862        1.031 N_0019_total       1
N_0019_total_cluster2     1   345136        0.878 N_0019_total       2
N_0019_total_cluster3     1   131336        0.971 N_0019_total       3
N_0019_total_cluster5     1   157127        1.083 N_0019_total       5
N_0019_total_cluster6     1    57861        1.285 N_0019_total       6
N_0021_total_cluster1     1   508776        0.870 N_0021_total       1
N_0021_total_cluster2     1    91463        0.930 N_0021_total       2
N_0064_total_cluster0     1   154672        1.071 N_0064_total       0
N_0064_total_cluster1     1   230306        1.064 N_0064_total       1

> summary(y$samples$norm.factors)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.673   0.915   1.026   1.009   1.113   1.285
```

4.10.4 Data exploration

We explore the pseudo-bulk profiles using a multi-dimensional scaling (MDS) plot. This visualizes the differences between the expression profiles of different samples in two dimensions. It can be seen that pseudo-bulk samples from the same cell cluster are close to each other.

```
> cluster <- as.factor(y$samples$cluster)
> plotMDS(y, pch=16, col=c(2:8)[cluster], main="MDS")
> legend("bottomleft", legend=paste0("cluster",levels(cluster)),
+     pch=16, col=2:8, cex=0.8)
```



4.10.5 Design matrix

To perform differential expression analysis between cell clusters, we create a design matrix using both cluster and donor information.

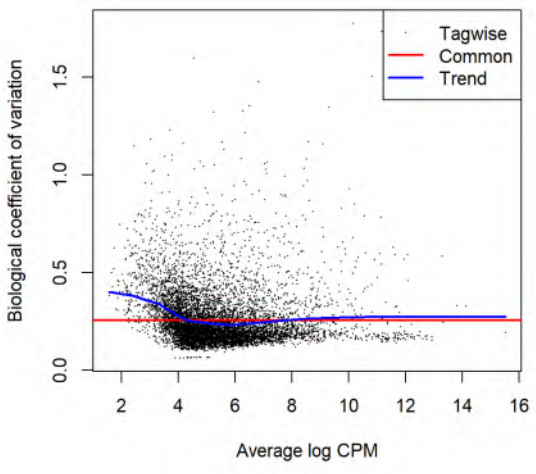
```
> donor <- factor(y$samples$sample)
> design <- model.matrix(~ cluster + donor)
> colnames(design) <- gsub("donor", "", colnames(design))
> colnames(design)[1] <- "Int"
> head(design)

  Int cluster1 cluster2 cluster3 cluster4 cluster5 cluster6 N_0021_total
1   1        0        0        0        0        0        0            0
2   1        1        0        0        0        0        0            0
3   1        0        1        0        0        0        0            0
4   1        0        0        1        0        0        0            0
5   1        0        0        0        0        1        0            0
6   1        0        0        0        0        0        1            0
  N_0064_total N_0092_total N_0093_total N_0123_total N_0169_total
1            0            0            0            0            0
2            0            0            0            0            0
3            0            0            0            0            0
4            0            0            0            0            0
5            0            0            0            0            0
6            0            0            0            0            0
  N_0230.17_total N_0233_total N_0275_total N_0288_total N_0342_total
1               0            0            0            0            0
```

```
2               0           0           0           0           0
3               0           0           0           0           0
4               0           0           0           0           0
5               0           0           0           0           0
6               0           0           0           0           0
  N_0372_total
1            0
2            0
3            0
4            0
5            0
6            0

> dim(design)

[1] 59 19
```

4.10.6 Dispersion estimation

The NB dispersion can be estimated using the `estimateDisp` function and visualized with `plotBCV`.

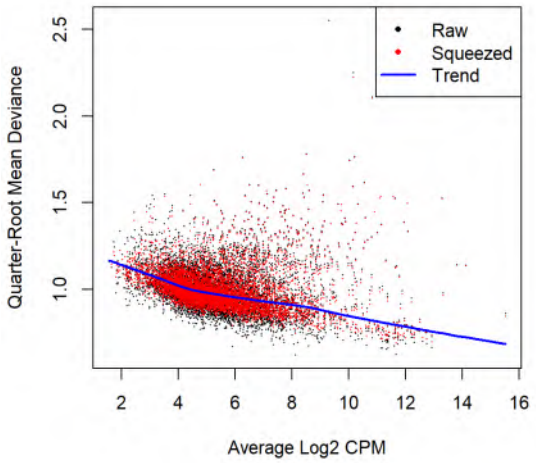
```
> y <- estimateDisp(y, design, robust=TRUE)
> y$common.dispersion

[1] 0.0651

> plotBCV(y)
```



Note that the NB dispersion estimates will not be used further under the latest quasi-likelihood (QL) pipeline.

The QL dispersions can be estimated using the `glmQLFit` function and visualized with `plotQLDisp`.

```
> fit <- glmQLFit(y, design, robust=TRUE)
> plotQLDisp(fit)
```



4.10.7 Marker genes identification

To confirm the identities of cell clusters, we perform differential expression analysis to identify marker genes of each cluster. In particular, we compare each cluster with all the other clusters. Since there are 7 clusters in total, we construct a contrast matrix as follows so that each column of the contrast matrix represents a testing contrast for one cell cluster.

```
> ncls <- nlevels(cluster)
> contr <- rbind( matrix(1/(1-ncls), ncls, ncls),
+                 matrix(0, ncol(design)-ncls, ncls) )
> diag(contr) <- 1
> contr[1,] <- 0
> rownames(contr) <- colnames(design)
> colnames(contr) <- paste0("cluster", levels(cluster))
> contr
```

```
               cluster0 cluster1 cluster2 cluster3 cluster4 cluster5 cluster6
Int               0.000    0.000    0.000    0.000    0.000    0.000    0.000
cluster1         -0.167    1.000   -0.167   -0.167   -0.167   -0.167   -0.167
cluster2         -0.167   -0.167    1.000   -0.167   -0.167   -0.167   -0.167
cluster3         -0.167   -0.167   -0.167    1.000   -0.167   -0.167   -0.167
cluster4         -0.167   -0.167   -0.167   -0.167    1.000   -0.167   -0.167
cluster5         -0.167   -0.167   -0.167   -0.167   -0.167    1.000   -0.167
cluster6         -0.167   -0.167   -0.167   -0.167   -0.167   -0.167    1.000
N_0021_total      0.000    0.000    0.000    0.000    0.000    0.000    0.000
N_0064_total      0.000    0.000    0.000    0.000    0.000    0.000    0.000
N_0092_total      0.000    0.000    0.000    0.000    0.000    0.000    0.000
N_0093_total      0.000    0.000    0.000    0.000    0.000    0.000    0.000
N_0123_total      0.000    0.000    0.000    0.000    0.000    0.000    0.000
```

```
N_0169_total       0.000    0.000    0.000    0.000    0.000    0.000    0.000
N_0230.17_total    0.000    0.000    0.000    0.000    0.000    0.000    0.000
N_0233_total       0.000    0.000    0.000    0.000    0.000    0.000    0.000
N_0275_total       0.000    0.000    0.000    0.000    0.000    0.000    0.000
N_0288_total       0.000    0.000    0.000    0.000    0.000    0.000    0.000
N_0342_total       0.000    0.000    0.000    0.000    0.000    0.000    0.000
N_0372_total       0.000    0.000    0.000    0.000    0.000    0.000    0.000
```

We then perform quasi-likelihood F-test for each testing contrast. The results are stored as a list of `DGELRT` objects, one for each comparison.

```
> qlf <- list()
> for(i in 1:ncls){
+     qlf[[i]] <- glmQLFTest(fit, contrast=contr[,i])
+     qlf[[i]]$comparison <- paste0("cluster", levels(cluster)[i], "_vs_others")
+ }
```

The top most significant DE genes of cluster 0 vs other clusters can be examined with `topTags`.

```
> topTags(qlf[[1]], n=10L)

Coefficient:  cluster0_vs_others
             gene logFC logCPM   F   PValue      FDR
FBLN1       FBLN1  6.01   6.78 709 2.03e-36 1.60e-32
OGN           OGN  5.74   5.84 586 6.24e-36 2.46e-32
CRABP2     CRABP2  3.95   6.35 436 5.93e-32 1.56e-28
SERPINF1 SERPINF1  5.17   6.92 580 1.07e-31 2.09e-28
MFAP4       MFAP4  4.62   5.95 433 1.87e-31 2.94e-28
IGFBP6     IGFBP6  5.37   6.79 497 9.57e-31 1.25e-27
GFPT2       GFPT2  3.50   6.43 435 1.37e-30 1.54e-27
LRP1         LRP1  4.35   5.59 393 2.26e-30 2.22e-27
MEG8         MEG8  4.29   5.41 371 4.61e-30 4.03e-27
DPT           DPT  5.93   6.31 415 2.55e-29 2.01e-26
```

The numbers of DE genes under each comparison are shown below.

```
> dt <- lapply(lapply(qlf, decideTests), summary)
> dt.all <- do.call("cbind", dt)
> dt.all

       cluster0_vs_others cluster1_vs_others cluster2_vs_others
Down                 1463                775               1442
NotSig               4007               4876               4298
Up                   2397               2216               2127
       cluster3_vs_others cluster4_vs_others cluster5_vs_others
Down                 1588               1608                243
NotSig               4423               4939               6596
Up                   1856               1320               1028
       cluster6_vs_others
Down                 1415
NotSig               4869
Up                   1583
```

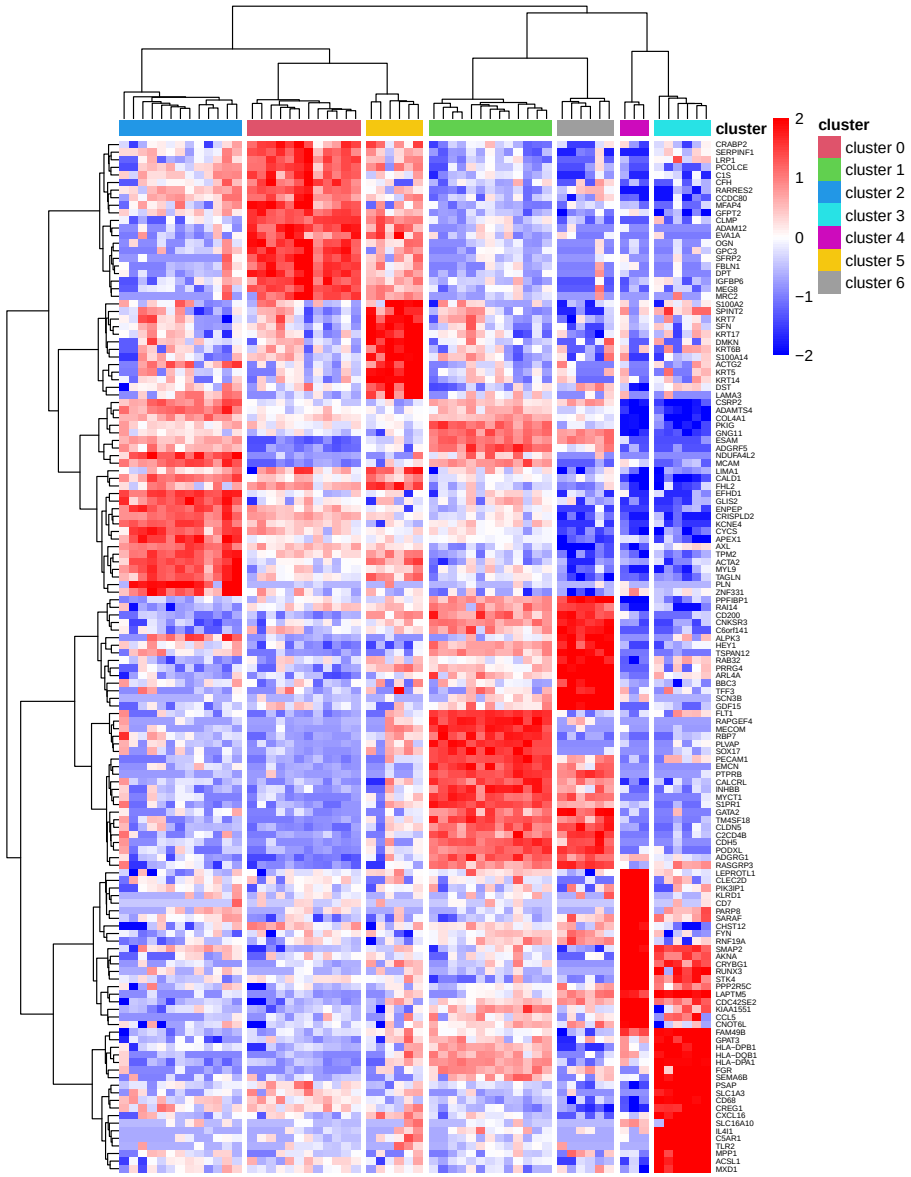
We select the top 20 marker (up-regulated) genes for each cluster.

```
> top <- 20
> topMarkers <- list()
> for(i in 1:ncls) {
+     ord <- order(qlf[[i]]$table$PValue, decreasing=FALSE)
+     up <- qlf[[i]]$table$logFC[ord] > 0
+     topMarkers[[i]] <- rownames(y)[ord[up][1:top]]
+ }
> topMarkers <- unique(unlist(topMarkers))
> topMarkers
```

```
  [1] "FBLN1"    "OGN"      "CRABP2"   "SERPINF1" "MFAP4"    "IGFBP6"
  [7] "GFPT2"    "LRP1"     "MEG8"     "DPT"      "MRC2"     "RARRES2"
 [13] "PCOLCE"   "SFRP2"    "GPC3"     "C1S"      "ADAM12"   "EVA1A"
 [19] "CFH"      "CCDC80"   "PLVAP"    "INHBB"    "EMCN"     "PECAM1"
 [25] "FLT1"     "RAPGEF4"  "ESAM"     "CDH5"     "RBP7"     "PTPRB"
 [31] "MECOM"    "ADGRF5"   "SOX17"    "CALCRL"   "PKIG"     "ADGRG1"
 [37] "TM4SF18"  "MYCT1"     "S1PR1"    "GNG11"    "TPM2"     "CRISPLD2"
 [43] "MYL9"     "CALD1"    "ACTA2"    "ADAMTS4"  "PLN"      "NDUFA4L2"
 [49] "MCAM"     "CYCS"     "EFHD1"    "TAGLN"    "CSRP2"    "ENPEP"
 [55] "APEX1"    "COL4A1"   "KCNE4"    "GLIS2"    "AXL"      "ZNF331"
 [61] "ACSL1"    "CD68"     "HLA-DQB1" "C5AR1"    "IL4I1"    "HLA-DPA1"
 [67] "CXCL16"   "MPP1"     "TLR2"     "PSAP"     "LAPTM5"   "HLA-DPB1"
 [73] "MXD1"     "FGR"      "SEMA6B"   "FAM49B"   "CREG1"    "GPAT3"
 [79] "SLC16A10" "SLC1A3"   "KLRD1"    "LEPROTL1" "PIK3IP1"  "CLEC2D"
 [85] "CCL5"     "KIAA1551" "PARP8"    "SARAF"    "AKNA"     "CRYBG1"
 [91] "CD7"      "RUNX3"    "PPP2R5C"  "SMAP2"    "FYN"      "CHST12"
 [97] "STK4"     "CNOT6L"   "RNF19A"   "CDC42SE2" "SFN"      "KRT5"
[103] "KRT17"    "KRT14"    "S100A2"   "DST"      "LAMA3"    "KRT6B"
[109] "S100A14"  "LIMA1"    "ACTG2"    "KRT7"     "FHL2"     "CLMP"
[115] "DMKN"     "SPINT2"   "CD200"    "PPFIBP1"  "SCN3B"    "CNKSR3"
[121] "GATA2"    "C2CD4B"   "GDF15"    "HEY1"     "TSPAN12"  "PRRG4"
[127] "BBC3"     "RAB32"    "CLDN5"    "ARL4A"    "C6orf141" "RASGRP3"
[133] "TFF3"     "ALPK3"    "RAI14"    "PODXL"
```

A heat map is produced to visualize the top marker genes across all the pseudo-bulk samples.

```
> lcpm <- cpm(y, log=TRUE)
> annot <- data.frame(cluster=paste0("cluster ", cluster))
> rownames(annot) <- colnames(y)
> ann_colors <- list(cluster=2:8)
> names(ann_colors$cluster) <- paste0("cluster ", levels(cluster))
> pheatmap::pheatmap(lcpm[topMarkers, ], breaks=seq(-2,2,length.out=101),
+     color=colorRampPalette(c("blue","white","red"))(100), scale="row",
+     cluster_cols=TRUE, border_color="NA", fontsize_row=5,
+     treeheight_row=70, treeheight_col=70, cutree_cols=7,
+     clustering_method="ward.D2", show_colnames=FALSE,
+     annotation_col=annot, annotation_colors=ann_colors)
```



4.10.8 Setup

This analysis was conducted using the following software setup:

```
> sessionInfo()

R version 4.4.0 (2024-04-24 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 10 x64 (build 19045)

Matrix products: default


locale:
```

```
[1] LC_COLLATE=English_Australia.utf8  LC_CTYPE=English_Australia.utf8
[3] LC_MONETARY=English_Australia.utf8 LC_NUMERIC=C
[5] LC_TIME=English_Australia.utf8

time zone: Australia/Sydney
tzcode source: internal

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

other attached packages:
[1] edgeR_4.1.32     limma_3.59.10    knitr_1.46       BiocStyle_2.31.0

loaded via a namespace (and not attached):
  [1] deldir_2.0-4           pbapply_1.7-2          gridExtra_2.3
  [4] rlang_1.1.3            magrittr_2.0.3         RcppAnnoy_0.0.22
  [7] spatstat.geom_3.2-9    matrixStats_1.3.0      ggridges_0.5.6
 [10] compiler_4.4.0         png_0.1-8              vctrs_0.6.5
 [13] reshape2_1.4.4         stringr_1.5.1          pkgconfig_2.0.3
 [16] fastmap_1.1.1          labeling_0.4.3         utf8_1.2.4
 [19] promises_1.3.0         rmarkdown_2.26         purrr_1.0.2
 [22] xfun_0.43              jsonlite_1.8.8         goftest_1.2-3
 [25] highr_0.10             later_1.3.2            spatstat.utils_3.0-4
 [28] irlba_2.3.5.1          parallel_4.4.0         cluster_2.1.6
 [31] R6_2.5.1               ica_1.0-3              spatstat.data_3.0-4
 [34] stringi_1.8.3          RColorBrewer_1.1-3     reticulate_1.36.1
 [37] parallelly_1.37.1      lmtest_0.9-40          scattermore_1.2
 [40] Rcpp_1.0.12            tensor_1.5             future.apply_1.11.2
 [43] zoo_1.8-12             sctransform_0.4.1      httpuv_1.6.15
 [46] Matrix_1.7-0           splines_4.4.0          igraph_2.0.3
 [49] tidyselect_1.2.1       abind_1.4-5            yaml_2.3.8
 [52] spatstat.random_3.2-3  spatstat.explore_3.2-7 codetools_0.2-20
 [55] miniUI_0.1.1.1         listenv_0.9.1          lattice_0.22-6
 [58] tibble_3.2.1           plyr_1.8.9             withr_3.0.0
 [61] shiny_1.8.1.1          ROCR_1.0-11            evaluate_0.23
 [64] Rtsne_0.17             future_1.33.2          fastDummies_1.7.3
 [67] survival_3.5-8         polyclip_1.10-6        fitdistrplus_1.1-11
 [70] pillar_1.9.0           BiocManager_1.30.22    Seurat_5.0.3
 [73] KernSmooth_2.23-22     plotly_4.10.4          generics_0.1.3
 [76] RcppHNSW_0.6.0         sp_2.1-3               ggplot2_3.5.1
 [79] munsell_0.5.1          scales_1.3.0           globals_0.16.3
 [82] xtable_1.8-4           glue_1.7.0             pheatmap_1.0.12
 [85] lazyeval_0.2.2         tools_4.4.0            data.table_1.15.4
 [88] RSpectra_0.16-1        locfit_1.5-9.9         RANN_2.6.1
 [91] leiden_0.4.3.1         dotCall64_1.1-1        cowplot_1.1.3
 [94] grid_4.4.0             tidyr_1.3.1            colorspace_2.1-0
 [97] nlme_3.1-164           patchwork_1.2.0        cli_3.6.2
[100] spatstat.sparse_3.0-3  spam_2.10-0            fansi_1.0.6
[103] viridisLite_0.4.2      dplyr_1.1.4            uwot_0.2.2
[106] gtable_0.3.5           digest_0.6.35          progressr_0.14.0
[109] ggrepel_0.9.5          farver_2.1.1           htmlwidgets_1.6.4
```

```
       [112] SeuratObject_5.0.1     htmltools_0.5.8.1      lifecycle_1.0.4
       [115] httr_1.4.7             statmod_1.5.0          mime_0.12
       [118] MASS_7.3-60.2
```

Bibliography

1. Baldoni, P.L., Chen, Y., Hediyeh-zadeh, S., Liao, Y., Dong, X., Rithie, M.E., Shi, W., and Smyth, G.K. (2024). Dividing out quantification uncertainty allows efficient assessment of differential transcript expression with edgeR. *Nucleic Acids Research* 52, e13.

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

3. Bray, N.L., Pimentel, H., Melsted, P., and Pachter, L. (2016). Near-optimal probabilistic RNA-seq quantification. *Nature biotechnology* 34, 525.

4. Chen, Y., Lun, A.T.L., and Smyth, G.K. (2014). Differential expression analysis of complex RNA-seq experiments using edgeR. In S. Datta and D.S. Nettleton, editors, *Statistical Analysis of Next Generation Sequence Data*, pages 51–74. Springer, New York.

5. Chen, Y., Pal, B., Lindeman, G.J., Visvader, J.E., and Smyth, G.K. (2022). R code and downstream analysis objects for the scRNA-seq atlas of normal and tumorigenic human breast tissue. *Scientific Data* 9, 96.

6. Chen, Y., Pal, B., Visvader, J.E., and Smyth, G.K. (2017). Differential methylation analysis of reduced representation bisulfite sequencing experiments using edgeR. *F1000Research* 6, 2055.

7. Chen, Y. and Smyth, G.K. (2021). Data, R code and output Seurat objects for single cell RNA-seq analysis of human breast tissues. figshare https://doi.org/10.6084/m9.figshare.17058077.

8. Cox, D.R. and Reid, N. (1987). Parameter orthogonality and approximate conditional inference. *Journal of the Royal Statistical Society: Series B (Methodological)* 49, 1–18.

9. Crowell, H.L., Soneson, C., Germain, P.L., Calini, D., Collin, L., Raposo, C., Malhotra, D., and Robinson, M.D. (2020). *muscat* detects subpopulation-specific state transitions from multi-sample multi-condition single-cell transcriptomics data. *Nature Communications* 11, 6077.

10. Cumbie, J.S., Kimbrel, J.A., Di, Y., Schafer, D.W., Wilhelm, L.J., Fox, S.E., Sullivan, C.M., Curzon, A.D., Carrington, J.C., Mockler, T.C., and Chang, J.H. (2011). GENE-Counter: A computational pipeline for the analysis of RNA-Seq data for gene expression differences. *PLOS ONE* 6, e25279.

11. Dai, Z., Sheridan, J.M., Gearing, L.J., Moore, D.L., Su, S., Wormald, S., Wilcox, S., O'Connor, L., Dickins, R.A., Blewitt, M.E., and Ritchie, M.E. (2014). edgeR: a versatile tool for the analysis of shRNA-seq and CRISPR-Cas9 genetic screens. *F1000Research* 3, 95.

12. Dong, X., Du, M.R.M., Gouil, Q., Tian, L., Jabbari, J.S., Bowden, R., Baldoni, P.L., Chen, Y., Smyth, G.K., Amarasinghe, S.L., Law, C.W., and Ritchie, M.E. (2023). Benchmarking long-read RNA-sequencing analysis tools using in silico mixtures. *Nature Methods* 20, 1810–1821.

13. Du, P., Zhang, X., Huang, C.C., Jafari, N., Kibbe, W.A., Hou, L., and Lin, S.M. (2010). Comparison of Beta-value and M-value methods for quantifying methylation levels by microarray analysis. *BMC Bioinformatics* 11, 587.

14. Dunn, P.K. and Smyth, G.K. (2018). *Generalized Linear Models With Examples in R*. Springer-Verlag, New York.

15. Frazee, A.C., Langmead, B., and Leek, J.T. (2011). ReCount: a multi-experiment resource of analysis-ready RNA-seq gene count datasets. *BMC Bioinformatics* 12, 449.

16. Fu, N.Y., Pal, B., Chen, Y., Jackling, F., Milevskiy, M., Vaillant, F., Capaldo, B., Guo, F., Liu, K.H., Rios, A.C., Lim, N., Kueh, A.J., Virshup, D.M., Herold, M.J., Tucker, H.O., Smyth, G.K., Lindeman, G.J., and Visvader, J.E. (2018). Foxp1 is indispensable for ductal morphogenesis and controls the exit of mammary stem cells from quiescence. *Developmental Cell* 47, 629–644.

17. Fu, N.Y., Rios, A.C., Pal, B., Soetanto, R., Lun, A.T.L., Liu, K., Beck, T., Best, S.A., Vaillant, F., Bouillet, P., Strasser, A., Preiss, T., Smyth, G.K., Lindeman, G., and Visvader, J. (2015). EGF-mediated induction of Mcl-1 at the switch to lactation is essential for alveolar cell survival. *Nature Cell Biology* 17, 365–375.

18. Gahurova, L., Tomizawa, S.i., Smallwood, S.A., Stewart-Morgan, K.R., Saadeh, H., Kim, J., Andrews, S.R., Chen, T., and Kelsey, G. (2017). Transcription and chromatin determinants of de novo DNA methylation timing in oocytes. *Epigenetics & Chromatin* 10, 25.

19. Graveley, B.R., Brooks, N., Carlson, J.W., Duff, M.O., Landolin, J.M., Yang, L., Artieri, G., van Baren, M.J., Boley, N., Booth, B.W., Brown, J.B., Cherbas, L., Davis, C.A., Dobin, A., Li, R., Lin, W., Malone, J.H., Mattiuzzo, N.R., Miller, D., Sturgill, D., Tuch, B.B., Zaleski, C., Zhang, D., Blanchette, M., and Dudoit, S. (2011). The developmental transcriptome of drosophila melanogaster. *Nature* 471, 473–479.

20. Hansen, K.D., Irizarry, R.A., and Wu, Z. (2012). Removing technical variability in RNA-seq data using conditional quantile normalization. *Biostatistics* 13, 204–216.

21. International HapMap Consortium, T. (2005). A haplotype map of the human genome. *Nature* 437, 1299–1320.

22. Krueger, F. and Andrews, S.R. (2011). Bismark: a flexible aligner and methylation caller for Bisulfite-Seq applications. *Bioinformatics* 27, 1571–1572.

23. Law, C.W., Chen, Y., Shi, W., and Smyth, G.K. (2014). Voom: precision weights unlock linear model analysis tools for RNA-seq read counts. *Genome Biology* 15, R29.

24. Liao, Y., Smyth, G.K., and Shi, W. (2013). The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. *Nucleic Acids Research* 41, e108.

25. Liao, Y., Smyth, G.K., and Shi, W. (2014). featureCounts: an efficient general-purpose read summarization program. *Bioinformatics* 30, 923–930.

26. Liao, Y., Smyth, G.K., and Shi, W. (2019). The R package Rsubread is easier, faster, cheaper and better for alignment and quantification of RNA sequencing reads. *Nucleic Acids Research* 47, e47.

27. Lun, A.T.L., Chen, Y., and Smyth, G.K. (2016). It's DE-licious: a recipe for differential expression analyses of RNA-seq experiments using quasi-likelihood methods in edgeR. *Methods in Molecular Biology* 1418, 391–416.

28. Lund, S.P., Nettleton, D., McCarthy, D.J., and Smyth, G.K. (2012). Detecting differential expression in RNA-sequence data using quasi-likelihood with shrunken dispersion estimates. *Statistical Applications in Genetics and Molecular Biology* 11, Article 8.

29. Marioni, J.C., Mason, C.E., Mane, S.M., Stephens, M., and Gilad, Y. (2008). RNA-seq: An assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research* 18, 1509–1517.

30. McCarthy, D.J., Chen, Y., and Smyth, G.K. (2012). Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. *Nucleic Acids Research* 40, 4288–4297.

31. McCarthy, D.J. and Smyth, G.K. (2009). Testing significance relative to a fold-change threshold is a TREAT. *Bioinformatics* 25, 765–771.

32. Meissner, A., Gnirke, A., Bell, G.W., Ramsahoye, B., Lander, E.S., and Jaenisch, R. (2005). Reduced representation bisulfite sequencing for comparative high-resolution DNA methylation analysis. *Nucleic acids research* 33, 5868–5877.

33. Pal, B., Chen, Y., Vaillant, F., Capaldo, B.D., Joyce, R., Song, X., Bryant, V., Penington, J.S., Di-Stefano, L., Ribera, N.T., Wilcox, S., Mann, G.B., kConFab, Papenfuss, A.T., Lindeman, G.J., Smyth, G.K., and Visvader, J.E. (2021). A single-cell RNA atlas of human breast spanning normal, preneoplastic and tumorigenic states. *EMBO Journal* 40, e107333.

34. Patro, R., Duggal, G., Love, M.I., Irizarry, R.A., and Kingsford, C. (2017). Salmon provides fast and bias-aware quantification of transcript expression. *Nature Methods* 14, 417.

35. Phipson, B., Lee, S., Majewski, I.J., Alexander, W.S., and Smyth, G.K. (2016). Robust hyperparameter estimation protects against hypervariable genes and improves power to detect differential expression. *Annals of Applied Statistics* 10, 946–963.

36. Pickrell, J.K., Marioni, J.C., Pai, A.A., Degner, J.F., Engelhardt, B.E., Nkadori, E., Veyrieras, J.B., Stephens, M., Gilad, Y., and Pritchard, J.K. (2010). Understanding mechanisms underlying human gene expression variation with RNA sequencing. *Nature* 464, 768–772.

37. Pickrell, J.K., Pai, A.A., Gilad, Y., and Pritchard, J.K. (2010). Noisy splicing drives mRNA isoform diversity in human cells. *PLoS Genetics* 6, e1001236.

38. Risso, D., Schwartz, K., Sherlock, G., and Dudoit, S. (2011). GC-content normalization for RNA-Seq data. *BMC Bioinformatics* 12, 480.

39. Robinson, M.D., McCarthy, D.J., and Smyth, G.K. (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* 26, 139–140.

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

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

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

43. Schübeler, D. (2015). Function and information content of DNA methylation. *Nature* 517, 321–326.

44. Shalem, O., Sanjana, N.E., Hartenian, E., Shi, X., Scott, D.A., Mikkelsen, T.S., Heckl, D., Ebert, B.L., Root, D.E., Doench, J.G., and Zhang, F. (2014). Genome-scale CRISPR-Cas9 knockout screening in human cells. *Science* 343, 84–7.

45. Smyth, G.K. (2004). Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology* 3, Article 3.

46. Stuart, T., Butler, A., Hoffman, P., Hafemeister, C., Papalexi, E., Mauck III, W.M., Hao, Y., Stoeckius, M., Smibert, P., and Satija, R. (2019). Comprehensive integration of single-cell data. *Cell* 177, 1888–1902.

47. Subramanian, A., Tamayo, P., Mootha, V.K., Mukherjee, S., Ebert, B.L., Gillette, M.A., Paulovich, A., Pomeroy, S.L., Golub, T.R., Lander, E.S., and Mesirov, J.P. (2005). Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci USA* 102, 15545–50.

48. Tuch, B.B., Laborde, R.R., Xu, X., Gu, J., Chung, C.B., Monighetti, C.K., Stanley, S.J., Olsen, K.D., Kasperbauer, J.L., Moore, E.J., Broomer, A.J., Tan, R., Brzoska, P.M., Muller, M.W., Siddiqui, A.S., Asmann, Y.W., Sun, Y., Kuersten, S., Barker, M.A., Vega, F.M.D.L., and Smith, D.I. (2010). Tumor transcriptome sequencing reveals allelic expression imbalances associated with copy number alterations. *PLoS ONE* 5, e9317.

49. Wu, D., Lim, E., Vaillant, F., Asselin-Labat, M., Visvader, J.E., and Smyth, G.K. (2010). ROAST: rotation gene set tests for complex microarray experiments. *Bioinformatics* 26, 2176–2182.

50. Wu, D. and Smyth, G.K. (2012). Camera: a competitive gene set test accounting for inter-gene correlation. *Nucleic Acids Research* 40, e133.