TCC: Differential expression analysis for tag count data with robust normalization strategies

Jianqiang Sun[1], Tomoaki Nishiyama[2§], Kentaro Shimizu[1], and Koji Kadota[1]

[1] The University of Tokyo, Tokyo, Japan

[2] Kanazawa University, Kanazawa, Japan

[§] Maintainer: Tomoaki Nishiyama (tomoakin at staff.kanazawa-u.ac.jp)

May 14, 2013

Abstract

The R/Bioconductor package, `TCC`, provides users with a robust and accurate framework to perform differential expression (DE) analysis of tag count data. We recently developed a multi-step normalization method (TbT; Kadota et al., 2012 [6]) for two-group RNA-seq data. The strategy (called DEGES) is to remove data that are potential differentially expressed genes (DEGs) before performing the data normalization. DEGES in `TCC` is essential for accurate normalization of tag count data, especially when the up- and down-regulated DEGs in one of the samples are extremely biased in their number. `TCC` provides a simple unified interface which encapsulates functions to calculate normalization factors and estimate DEGs defined in `edgeR`, `DESeq`, and `baySeq`. The appropriate combination provided by `TCC` allows a more robust and accurate estimation to be performed more easily than directly using original packages. Functions to produce simulation data under various conditions and to plot the data are also provided.

Contents

1 Introduction

Differential expression analysis based on tag count data has become a fundamental task for identifying differentially expressed genes or transcripts (DEGs). The TCC package (Tag Count Comparison; Sun et al., submitted [10]) provides users with a robust and accurate framework to perform differential expression analysis of tag count data. TCC provides integrated analysis pipelines with improved data normalization steps, compared with other packages such as edgeR, DESeq, and baySeq, by appropriately combining their functionalities. The package incorporates multi-step normalization methods whose strategy is to remove data that are potential DEGs before performing the data normalization.

Kadota et al. (2012) [6] recently reported that the normalization methods implemented in R packages (such as edgeR (Robinson et al., 2010 [7]), DESeq (Anders and Huber, 2010 [1]), and baySeq (Hardcastle and Kelly, 2010 [5])) for differential expression (DE) analysis between samples are inadequate when the up- and down-regulated DEGs in one of the samples are extremely biased in their number (i.e., biased DE). This is because the current methods implicitly assume a balanced DE, wherein the numbers of highly and lowly expressed DE entities in samples are (nearly) equal. As a result, methods assuming unbiased DE will not work well on data with biased DE. Although a major purpose of data normalization is to detect such DE entities, their existence themselves consequently interferes with their opportunity to be top-ranked. Conventional procedures for identifying DEGs from tag count data consisting of two steps (i.e., data normalization and identification of DEGs) cannot in principle eliminate the potential DE entities before data normalization.

To normalize data that potentially has various scenarios (including unbiased and biased DE), we recently proposed a multi-step normalization strategy (called TbT, an acronym for the TMM-baySeq-TMM pipeline; Kadota et al., 2012 [6]), in which the TMM normalization method (Robinson and Oshlack, 2010 [8]) is used in steps 1 and 3 and an empirical Bayesian method implemented in the baySeq package (Hardcastle and Kelly, 2010 [5]) is used in step 2. Although this multi-step DEG elimination strategy (called "DEGES" for short) can successfully remove potential DE entities identified in step 2 prior to the estimation of the normalization factors using the TMM normalization method in step 3, the baySeq package used in step 2 of the TbT method is much more computationally intensive than competing packages like edgeR and DESeq. While the three-step TbT normalization method performed best on simulated and real tag count data, it is practically possible to make different choices for the methods in each step. A more comprehensive study regarding better choices for DEGES is needed.

This package provides tools to perform multi-step normalization methods based on DEGES and enables differential expression analysis of tag count data without having to worry much about biased distributions of DEGs. The DEGES-based normalization function implemented in TCC includes the TbT method based on DEGES for two-group data with or without replicates, much faster method, and methods for multi-group comparison. TCC provides a simple unified interface to perform data normalization with combinations of functions provided by baySeq, DESeq, and edgeR. Functions to produce simulation data under various conditions and to plot the data are also provided.

1.1 Installation

This package is available from the Bioconductor website (http://bioconductor.org/). To install the package, enter the following command after starting R:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("TCC")
```

1.2 Citations

This package internally uses many of the functions implemented in the other packages. This is because our normalization procedures consist, in part, of combinations of existing normalization methods and differential expression (DE) methods.

For example, the TbT normalization method (Kadota et al., 2012 [6]), which is a particular functionality of the `TCC` package (Sun et al., submitted [10]), consists of the TMM normalization method (Robinson and Oshlack, 2010 [8]) implemented in the `edgeR` package (Robinson et al., 2010 [7]) and the empirical Bayesian method implemented in the `baySeq` package (Hardcastle and Kelly, 2010 [5]). Therefore, please cite the appropriate references when you publish your results.

```
> citation("TCC")
```

```
Please cite appropriate references when you publish your results.

  Anders S and Huber W. Differential expression analysis for sequence
  count data. Genome Biol. 11(10): R106, 2010

  Di Y, Schafer DW, Cumbie JS, and Chang JH. The NBP negative binomial
  model for assessing differential gene expression from RNA-Seq. Stat
  Appl Genet Mol Biol. 10: art24, 2011

  Hardcastle TJ and Kelly KA. baySeq: empirical Bayesian methods for
  identifying differential expression in sequence count data. BMC
  Bioinformatics 11: 422, 2010

  Kadota K, Nishiyama T, and Shimizu K. A normalization strategy for
  comparing tag count data. Algorithms Mol Biol. 7:5, 2012

  Robinson MD, McCarthy DJ, and Smyth GK. edgeR: a Bioconductor package
  for differential expression analysis of digital gene expression data.
  Bioinformatics 26(1): 139-140, 2010

  Robinson MD and Oshlack A. A scaling normalization method for
  differential expression analysis of RNA-seq data. Genome Biol. 11:
  R25, 2010

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

  Sun J, Nishiyama T, Shimizu K, and Kadota K. TCC: an R package for
  comparing tag count data with robust normalization strategies.
  submitted

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

1.3 Quick start

Let us begin by showing two examples (Cases1 and 2) of identifying DEGs between two groups from tag count data consisting of 1,000 genes and a total of six samples (each group has three biological replicates). The hypothetical count data (termed "`hypoData`") is stored in this package (for details, see section 2.1). We then describe the DE analysis of count data without replicates (i.e., two samples), using the data of the first and the fourth column of `hypoData` (Case 3). We recommend the use of commands in Cases 2 and 3.

Case 1: DE analysis of two-group count data with replicates by using the exact test (Robinson and Smyth, 2008 [9]) in `edgeR` coupled with TbT normalization (termed the TbT-`edgeR` combination). The `TCC` package was originally designed with the TbT normalization method, and the original study (Kadota et al., 2012 [6]) recommended this analysis pipeline. Note that a smaller sampling size (i.e., `samplesize = 100`) is used here to reduce the computation time, but a larger sampling size of around 10,000 (i.e., `samplesize = 10000`) is recommended (Hardcastle and Kelly, 2010 [5]). Suggested citations are as follows: `TCC` (Sun et al., submitted [10]), TbT (Kadota et al., 2012 [6]), TMM (Robinson and Oshlack, 2010 [8]), `baySeq` (Hardcastle and Kelly, 2010 [5]), and `edgeR` (Robinson et al., 2010 [7]). For details, see section 3.1.1.

```
> library(TCC)
> data(hypoData)
> samplesize <- 100
> group <- c(1, 1, 1, 2, 2, 2)
> tcc <- new("TCC", hypoData, group)
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "bayseq",
+                        iteration = 1, samplesize = samplesize)
> tcc <- estimateDE(tcc, test.method = "edger", FDR = 0.1)
> result <- getResult(tcc, sort = TRUE)
> head(result)
```

```
     gene_id  a.value   m.value      p.value      q.value rank estimatedDEG
151 gene_151 9.736778 -2.746541 5.087098e-11 5.087098e-08    1            1
599 gene_599 5.925872 -3.277967 9.784688e-10 3.261577e-07    2            1
39   gene_39 7.111121 -2.453829 9.784732e-10 3.261577e-07    3            1
68   gene_68 6.209070 -2.862546 4.880368e-09 1.220092e-06    4            1
144 gene_144 7.588227 -2.122883 1.242359e-08 2.087463e-06    5            1
175 gene_175 7.984823 -2.364720 1.252478e-08 2.087463e-06    6            1
```

Case 2: DE analysis for two-group count data with replicates by using the exact test coupled with iterative DEGES/`edgeR` normalization (i.e., the iDEGES/`edgeR`-`edgeR` combination). This is an alternative pipeline designed to reduce the runtime (approx. 20 sec.), yet its performance is comparable to the above pipeline. Accordingly, we recommend using this pipeline as a default when analyzing tag count data with replicates. A notable advantage of this pipeline is that the multi-step normalization strategy only needs the methods implemented in the `edgeR` package. The suggested citations are as follows: `TCC` (Sun et al., submitted [10]), TMM (Robinson and Oshlack, 2010 [8]), the exact test (Robinson and Smyth, 2008 [9]), and `edgeR` (Robinson et al., 2010 [7]). For details, see section 3.1.3.

```
> library(TCC)
> data(hypoData)
> group <- c(1, 1, 1, 2, 2, 2)
```

```
> tcc <- new("TCC", hypoData, group)
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 3, FDR = 0.1, floorPDEG = 0.05)
> tcc <- estimateDE(tcc, test.method = "edger", FDR = 0.1)
> result <- getResult(tcc, sort = TRUE)
> head(result)
```

```
     gene_id  a.value   m.value      p.value      q.value rank estimatedDEG
151 gene_151 9.736785 -2.753816 4.641083e-11 4.641083e-08    1            1
39   gene_39 7.110842 -2.460691 9.115042e-10 3.270783e-07    2            1
599 gene_599 5.927173 -3.282264 9.812348e-10 3.270783e-07    3            1
68   gene_68 6.209395 -2.867694 4.776945e-09 1.194236e-06    4            1
175 gene_175 7.984265 -2.373657 1.082493e-08 1.899827e-06    5            1
144 gene_144 7.588164 -2.130092 1.139896e-08 1.899827e-06    6            1
```

Case 3: DE analysis for two-group count data without replicates by using the negative binomial (NB) test in `DESeq` coupled with iDEGES/`DESeq` normalization (i.e., the iDEGES/`DESeq`-`DESeq` combination). A procedure using the data of the first and fourth columns of `hypoData` is shown here. Similar to Case 2, this pipeline entirely consists of methods implemented in the `DESeq` package. Suggested citations are as follows: `TCC` (Sun et al., submitted [10]) and `DESeq` (Anders and Huber, 2010 [1]). For details, see section 3.2.

```
> library(TCC)
> data(hypoData)
> group <- c(1, 2)
> tcc <- new("TCC", hypoData[,c(1,4)], group)
> tcc <- calcNormFactors(tcc, norm.method = "deseq", test.method = "deseq",
+                        iteration = 3, FDR = 0.1, floorPDEG = 0.05)
> tcc <- estimateDE(tcc, test.method = "deseq", FDR = 0.1)
> result <- getResult(tcc, sort = TRUE)
> head(result)
```

```
     gene_id    a.value   m.value      p.value   q.value rank estimatedDEG
36   gene_36 -0.9988563 -8.525340 0.0002119999 0.2011879    1            0
17   gene_17  5.9635499 -5.470058 0.0064590145 1.0000000    2            0
5     gene_5  3.3084986 -6.499805 0.0184836363 1.0000000    3            0
989 gene_989 -0.9988563 -5.839291 0.0252092271 1.0000000    4            0
187 gene_187  3.6973024  5.507222 0.0321082546 1.0000000    5            0
822 gene_822 -0.9988563  5.436833 0.0522752310 1.0000000    6            0
```

2 Preparations

2.1 Reading the count data

Similar to the other packages, TCC typically starts the DE analysis with a count table matrix where each row indicates a gene (or transcript), each column indicates a sample (or library), and each cell indicates the number of counts for a gene in a sample. Here, we assume a hypothetical count matrix consisting of 1,000 rows (or genes) and a total of six columns (the first three columns are produced from biological replicates of Group 1 and the remaining columns are from Group 2); i.e., {G1_rep1, G1_rep2, G1_rep3} vs. {G2_rep1, G2_rep2, G2_rep3}. We start by loading the hypothetical data (hypoData) from TCC and giving a numeric vector (group) indicating which group each sample belongs to.

```
> library(TCC)
> data(hypoData)
> head(hypoData)
```

```
       G1_rep1 G1_rep2 G1_rep3 G2_rep1 G2_rep2 G2_rep3
gene_1      34      45     122      16      14      29
gene_2     358     388      22      36      25      68
gene_3    1144     919     990     374     480     239
gene_4       0       0      44      18       0       0
gene_5      98      48      17       1       8       5
gene_6     296     282     216      86      62      69
```

```
> dim(hypoData)
```

```
[1] 1000    6
```

```
> group <- c(1, 1, 1, 2, 2, 2)
```

If you want to analyze another count matrix consisting of nine columns (e.g., the first four columns are produced from biological replicates of G1, and the remaining five columns are from G2), the group vector should be indicated as follows.

```
> group <- c(1, 1, 1, 1, 2, 2, 2, 2, 2)
```

2.2 Constructing TCC class object

The new function has to be used to perform the main functionalities of TCC. This function constructs a TCC class object, and subsequent analyses are performed on this class object. The object is constructed from i) a count matrix (hypoData) and ii) the corresponding numeric vector (group) as follows.

```
> library(TCC)
> data(hypoData)
> group <- c(1, 1, 1, 2, 2, 2)
> tcc <- new("TCC", hypoData, group)
> tcc
```

```
Count:
       G1_rep1 G1_rep2 G1_rep3 G2_rep1 G2_rep2 G2_rep3
gene_1      34      45     122      16      14      29
gene_2     358     388      22      36      25      68
gene_3    1144     919     990     374     480     239
gene_4       0       0      44      18       0       0
gene_5      98      48      17       1       8       5
gene_6     296     282     216      86      62      69

Sample:
        group norm.factors lib.sizes
G1_rep1     1            1    142177
G1_rep2     1            1    145289
G1_rep3     1            1    149886
G2_rep1     2            1    112100
G2_rep2     2            1    104107
G2_rep3     2            1    101975
```

The count matrix and group vector information can be retrieved from the stored class object by using `tcc$count` and `tcc$group`, respectively.

```
> head(tcc$count)
```

```
       G1_rep1 G1_rep2 G1_rep3 G2_rep1 G2_rep2 G2_rep3
gene_1      34      45     122      16      14      29
gene_2     358     388      22      36      25      68
gene_3    1144     919     990     374     480     239
gene_4       0       0      44      18       0       0
gene_5      98      48      17       1       8       5
gene_6     296     282     216      86      62      69
```

```
> tcc$group
```

```
        group
G1_rep1     1
G1_rep2     1
G1_rep3     1
G2_rep1     2
G2_rep2     2
G2_rep3     2
```

The subset of `TCC` class object can be taken by the `subset` or `"["` functions.

```
> dim(tcc$count)
```

```
[1] 1000    6
```

```
> tcc.sub1 <- subset(tcc, c(rep(TRUE, 20), rep(FALSE, 980)))
> dim(tcc.sub1$count)
```

```
[1] 20  6
```

```
> tcc.sub2 <- tcc[1:20]
> dim(tcc.sub2$count)
```

```
[1] 20  6
```

2.3 Filtering low-count genes (optional)

The way to filter out genes with low-count tags across samples depends on the user's philosophy. Although we recommend removing tags with zero counts across samples as a minimum filtering, this effort is optional. The `filterLowCountGenes` function performs this filtering.

```
> library(TCC)
> data(hypoData)
> group <- c(1, 1, 1, 2, 2, 2)
> tcc <- new("TCC", hypoData, group)
> tcc <- filterLowCountGenes(tcc)
> dim(tcc$count)
```

```
[1] 996   6
```

It can be seen that $4 (= 1000 - 996)$ genes were filtered as non-expressed. The same procedure can be performed without the `filterLowCountGenes` function, in which case the filtering is performed before the `TCC` class object is constructed.

```
> filter <- as.logical(rowSums(hypoData) > 0)
> dim(hypoData[filter, ])
```

```
[1] 996   6
```

```
> tcc <- new("TCC", hypoData[filter, ], group)
> dim(tcc$count)
```

```
[1] 996   6
```

3 Normalization

3.1 Normalization of two-group count data with replicates

This package provides robust normalization methods based on DEGES proposed by Kadota et al. (2012) [6]. When obtaining normalization factors from two-group data with replicates, users can select a total of six combinations (two normalization methods $\times$ three DEG identification methods) coupled with an arbitrary number of iterations ($n = 0, 1, 2, \ldots, 100$) in our DEGES-based normalization pipeline. We show some of the practical combinations below.

Since the three-step TbT normalization method was originally designed for normalizing tag count data with (biological) replicates, we will first explain the TbT method (3.1.1 DEGES/TbT). In relation to the other DEGES-based methods, we will call the method "DEGES/TbT" for convenience. As mentioned in the original study, DEGES/TbT needs a long computation time. Accordingly, we present three shorter alternatives (3.1.2 DEGES/`edgeR`, 3.1.3 iDEGES/`edgeR`, and 3.1.4 DEGES/`DESeq`). Note that the purpose here is to obtain accurate normalization factors to be used with statistical models (e.g., the exact test or empirical Bayes) for the DE analysis described in the next section (4 **Differential expression**).

3.1.1 DEGES/TbT

The DEGES/TbT (Kadota et al., 2012 [6]) with default parameter settings can be performed as follows.

```
> set.seed(1000)
> library(TCC)
> data(hypoData)
> samplesize <- 70
> group <- c(1, 1, 1, 2, 2, 2)
> tcc <- new("TCC", hypoData, group)
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "bayseq",
+                        iteration = 1, samplesize = samplesize)
```

Note that a smaller sampling size (i.e., `samplesize = 70`) is used here to reduce the computation time when performing the empirical Bayesian method in step 2, but a larger sampling size of around $10,000$ (i.e., `samplesize = 10000`) is recommended (Hardcastle and Kelly, 2010 [5]). This method estimates an empirical distribution of the parameters of the NB distribution by bootstrapping from the input data. While the sampling size can be made smaller to reduce the computation time (e.g., `samplesize = 40`), the resulting normalization factors will vary from trial to trial. In this vignette, we will call the `set.seed` function for obtaining reproducible results (i.e., the `tcc$norm.factors` values) when using any random function. The calculated normalization factors and the computation time can be retrieved with the following commands.

```
> tcc$norm.factors
```

```
  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3
0.8890384 0.8581482 0.8393570 1.0803897 1.1345506 1.1985161
```

```
> tcc$DEGES$execution.time
```

```
   user  system elapsed
  5.348   0.008   5.373
```

Of course, the procedure can be performed by using functions in `edgeR` and `baySeq`, instead of using the `calcNormFators` function in `TCC`. The `calcNormFators` function together with the above parameter settings can be regarded as a wrapper function for the following commands.

```
> set.seed(1000)
> library(TCC)
> data(hypoData)
> samplesize <- 70
> group <- c(1, 1, 1, 2, 2, 2)
> ### STEP 1 ###
> d <- DGEList(count = hypoData, group = group)
> d <- calcNormFactors(d)
> norm.factors <- d$samples$norm.factors
> norm.factors <- norm.factors / mean(norm.factors)
> ### STEP 2 ###
> cD <- new("countData", data = hypoData, replicates = group,
+           groups = list(NDE = rep(1, length = length(group)), DE = group),
+           libsizes = colSums(hypoData) * norm.factors)
> cD <- getPriors.NB(cD, samplesize = samplesize, estimation = "QL", cl = NULL)
> cD <- getLikelihoods.NB(cD, pET = "BIC", cl = NULL)

.

> is.DEG <- as.logical(rank(-cD@posteriors[, "DE"]) <
+                        (nrow(hypoData) * cD@estProps[2]))
> ### STEP 3 ###
> d <- DGEList(count = hypoData[!is.DEG, ], group = group)
> d <- calcNormFactors(d)
> norm.factors <- d$samples$norm.factors * colSums(hypoData[!is.DEG, ]) /
+                 colSums(hypoData)
> norm.factors <- norm.factors / mean(norm.factors)
> norm.factors

  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3
0.8890384 0.8581482 0.8393570 1.0803897 1.1345506 1.1985161
```

3.1.2 DEGES/edgeR

Now let us describe an alternative approach that is roughly 200-400 times faster than DEGES/TbT, yet has comparable performance. The TMM-`edgeR`-TMM pipeline (called DEGES/`edgeR`) employs the exact test implemented in `edgeR` in step 2. To use this pipeline, we have to provide a reasonable threshold for defining potential DEGs in step 2. We will define the threshold as an arbitrary false discovery rate (FDR) with a floor value of P_{DEG}. The default FDR is < 0.1, and the default floor P_{DEG} is 5%, but different choices are of course possible. For example, in case of the default settings, $x\%(x > 5\%)$ of the top-ranked potential DEGs are eliminated in step 2 if the percentage $(= x\%)$ of genes satisfying FDR < 0.1 is over 5%. The DEGES/`edgeR` pipeline has an apparent advantage over TbT in computation time. It can be performed as follows:

```
> library(TCC)
> data(hypoData)
> group <- c(1, 1, 1, 2, 2, 2)
> tcc <- new("TCC", hypoData, group)
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 1, FDR = 0.1, floorPDEG = 0.05)
> tcc$norm.factors


  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3
0.8745111 0.8449577 0.8406663 1.0806355 1.1514213 1.2078082


> tcc$DEGES$execution.time


   user  system elapsed
  0.408   0.000   0.408
```

The normalization factors calculated from the DEGES/`edgeR` are very similar to those of DEGES/TbT with the default parameter settings (i.e., `samplesize = 10000`). For `edgeR` users, we provide commands, consisting of functions in `edgeR`, to perform the DEGES/`edgeR` pipeline without `TCC`. The `calcNormFators` function together with the above parameter settings can be regarded as a wrapper function for the following commands.

```
> library(TCC)
> data(hypoData)
> group <- c(1, 1, 1, 2, 2, 2)
> FDR <- 0.1
> floorPDEG <- 0.05
> d <- DGEList(counts = hypoData, group = group)
> ### STEP 1 ###
> d <- calcNormFactors(d)
> ### STEP 2 ###
> d <- estimateCommonDisp(d)
> d <- estimateTagwiseDisp(d)
> result <- exactTest(d)
> q.value <- p.adjust(result$table$PValue, method = "BH")
> if (sum(q.value < FDR) > (floorPDEG * nrow(hypoData))) {
+   is.DEG <- as.logical(q.value < FDR)
+ } else {
+   is.DEG <- as.logical(rank(result$table$PValue, ties.method = "min") <=
+                        nrow(hypoData) * floorPDEG)
+ }
> ### STEP 3 ###
> d <- DGEList(counts = hypoData[!is.DEG, ], group = group)
> d <- calcNormFactors(d)
> norm.factors <- d$samples$norm.factors * colSums(hypoData[!is.DEG, ]) /
+                   colSums(hypoData)
> norm.factors <- norm.factors / mean(norm.factors)
> norm.factors
```

```
  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3
0.8745111 0.8449577 0.8406663 1.0806355 1.1514213 1.2078082
```

3.1.3 iDEGES/edgeR

Our multi-step normalization can be repeated until the calculated normalization factors converge (Kadota et al., 2012 [6]). An iterative version of DEGES/TbT (i.e., iDEGES/TbT) can be described as the TMM-(baySeq-TMM)$_n$ pipeline with $n \geq 2$. Although the iDEGES/TbT would not be practical in terms of the computation time, the TMM-(edgeR-TMM)$_n$ pipeline (iDEGES/edgeR) is potentially superior to both the DEGES/edgeR and the DEGES/TbT. A suggested iDEGES/edgeR implementation ($n = 3$) consists of seven steps, as follows:

```
> library(TCC)
> data(hypoData)
> group <- c(1, 1, 1, 2, 2, 2)
> tcc <- new("TCC", hypoData, group)
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 3, FDR = 0.1, floorPDEG = 0.05)
> tcc$norm.factors
```

```
  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3
0.8766053 0.8450605 0.8346595 1.0842097 1.1538160 1.2056491
```

```
> tcc$DEGES$execution.time
```

```
   user  system elapsed
  1.040   0.000   1.047
```

3.1.4 DEGES/DESeq

The DEGES pipeline can also be performed by using only the functions in the DESeq package. Similar to the edgeR case above, this DESeq-DESeq-DESeq pipeline (DEGES/DESeq) changes the corresponding arguments of the norm.method and test.method as follows:

```
> library(TCC)
> data(hypoData)
> group <- c(1, 1, 1, 2, 2, 2)
> tcc <- new("TCC", hypoData, group)
> tcc <- calcNormFactors(tcc, norm.method = "deseq", test.method = "deseq",
+                        iteration = 1, FDR = 0.1, floorPDEG = 0.05)
> tcc$norm.factors
```

```
  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3
0.8885503 0.8810866 0.8298458 1.0698392 1.1513431 1.1793351
```

```
> tcc$DEGES$execution.time
```

```
   user  system elapsed
  0.688   0.000   0.694
```

For DESeq users, we also provide commands, consisting of functions in DESeq, to perform the DEGES/DESeq pipeline without TCC. The calcNormFators function together with the above arguments can be regarded as a wrapper function for the following commands.

```
> library(TCC)
> data(hypoData)
> group <- c(1, 1, 1, 2, 2, 2)
> FDR <- 0.1
> floorPDEG <- 0.05
> cds <- newCountDataSet(hypoData, group)
> ### STEP 1 ###
> cds <- estimateSizeFactors(cds)
> ### STEP 2 ###
> cds <- estimateDispersions(cds)
> result <- nbinomTest(cds, 1, 2)
> result$pval[is.na(result$pval)] <- 1
> result$padj[is.na(result$padj)] <- 1
> q.value <- result$padj
> if (sum(q.value < FDR) > (floorPDEG * nrow(hypoData))) {
+   is.DEG <- as.logical(q.value < FDR)
+ } else {
+   is.DEG <- as.logical(rank(result$pval, ties.method = "min") <=
+                        nrow(hypoData) * floorPDEG)
+ }
> ### STEP 3 ###
> cds <- newCountDataSet(hypoData[!is.DEG, ], group)
> cds <- estimateSizeFactors(cds)
> norm.factors <- sizeFactors(cds) / colSums(hypoData)
> norm.factors <- norm.factors / mean(norm.factors)
> norm.factors

  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3
0.8885503 0.8810866 0.8298458 1.0698392 1.1513431 1.1793351
```

3.2 Normalization of two-group count data without replicates

It is important to keep in mind that most R packages (including edgeR, DESeq, and baySeq) are primarily for analyzing data including biological replications because the biological variability has to be accurately estimated to avoid spurious DE calls (Glaus et al., 2012 [4]). In fact, the functions for the DEG identification method implemented in edgeR (i.e., the exact test; ver. 3.0.4) do not allow analysis without replicates, though the TMM normalization method in the package can be applied to data regardless of whether it has replicates. Although the edgeR manual provides users with some ideas on how to perform the DE analysis, it is difficult to customize the analysis with DEGES to data without replicates.

When obtaining normalization factors from two-group count data without replicates, users can select a total of four combinations (two normalization methods $\times$ two DEG identification

methods) coupled with an arbitrary number of iterations ($n = 0, 1, 2, \ldots, 100$) in our DEGES-based normalization pipeline. That is, the `calcNormFators` function with the `norm.method = "deseq"` or `"tmm"` and `test.method = "deseq"` or `"bayseq"` can be indicated. Let us explain the procedure by retrieving the data of the first and the fourth columns of `hypoData`, i.e.,

```
> library(TCC)
> data(hypoData)
> group <- c(1, 2)
> tcc <- new("TCC", hypoData[, c(1, 4)], group)
> head(tcc$count)
```

```
       G1_rep1 G2_rep1
gene_1      34      16
gene_2     358      36
gene_3    1144     374
gene_4       0      18
gene_5      98       1
gene_6     296      86
```

```
> tcc$group
```

```
        group
G1_rep1     1
G2_rep1     2
```

A DEGES pipeline (DEGES/`DESeq`) for obtaining normalization factors is as follows.

```
> tcc <- calcNormFactors(tcc, norm.method = "deseq", test.method = "deseq",
+                        iteration = 1, FDR = 0.1, floorPDEG = 0.05)
> tcc$norm.factors
```

```
 G1_rep1  G2_rep1
0.921658 1.078342
```

An advantage of this DEGES/`DESeq` pipeline is that the multi-step normalization strategy only needs the methods in the `DESeq` package. These factors should be the same as those produced by the following procedure consisting of functions implemented in `DESeq`.

```
> library(TCC)
> data(hypoData)
> group <- c(1, 2)
> FDR <- 0.1
> floorPDEG <- 0.05
> cds <- newCountDataSet(hypoData[, c(1, 4)], group)
> ### STEP 1 ###
> cds <- estimateSizeFactors(cds)
> ### STEP 2 ###
> cds <- estimateDispersions(cds, method = "blind", sharingMode = "fit-only")
```

```
> result <- nbinomTest(cds, 1, 2)
> result$pval[is.na(result$pval)] <- 1
> result$padj[is.na(result$padj)] <- 1
> q.value <- result$padj
> if (sum(q.value < FDR) > (floorPDEG * nrow(hypoData))) {
+   is.DEG <- as.logical(q.value < FDR)
+ } else {
+   is.DEG <- as.logical(rank(result$pval, ties.method = "min") <=
+                        nrow(hypoData) * floorPDEG)
+ }
> ### STEP 3 ###
> cds <- newCountDataSet(hypoData[!is.DEG, c(1, 4)], group)
> cds <- estimateSizeFactors(cds)
> norm.factors <- sizeFactors(cds) / colSums(hypoData[, c(1, 4)])
> norm.factors <- norm.factors / mean(norm.factors)
> norm.factors
```

```
 G1_rep1  G2_rep1
0.921658 1.078342
```

3.3 Normalization of multi-group count data with replicates

Many R packages (including `edgeR`, `DESeq`, and `baySeq`) support DE analysis for multi-group tag count data. `TCC` provides some prototypes of DEGES-based pipelines for such data. Here, we analyze another hypothetical three-group count matrix, the `hypoData_mg` object, provided in `TCC`. It consists of 1,000 genes and a total of nine columns for testing any difference among three groups that each have triplicates.

```
> library(TCC)
> data(hypoData_mg)
> group <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
> tcc <- new("TCC", hypoData_mg, group)
> tcc
```

```
Count:
       G1_rep1 G1_rep2 G1_rep3 G2_rep1 G2_rep2 G2_rep3 G3_rep1 G3_rep2 G3_rep3
gene_1      63      48      31      15      12      12      24      15      14
gene_2      18       0       7       2       3       8       3       5       2
gene_3     106      66      25       9      14      14      11      11       3
gene_4       4       9       6       1       6       1       0       2       2
gene_5       0       1       2       1       0       1       0       0       1
gene_6      57     100      83      20       5      16      26       7      21

Sample:
        group norm.factors lib.sizes
G1_rep1     1            1    150490
G1_rep2     1            1    166665
G1_rep3     1            1    199283
```

```
G2_rep1     2            1    183116
G2_rep2     2            1    126651
G2_rep3     2            1    131377
G3_rep1     3            1    149828
G3_rep2     3            1    150288
G3_rep3     3            1    141702
```

```
> dim(tcc$count)
```

```
[1] 1000    9
```

Of the 1,000 genes, the first 200 genes are DEGs and the remaining 800 genes are non-DEGs. The breakdowns for the 200 DEGs are as follows: 140, 40, and 20 DEGs are up-regulated in Groups 1, 2, and 3. Below, we show some DEGES-based normalization pipelines for this multi-group data (3.3.1 DEGES/TbT, 3.3.2 DEGES/`edgeR`, and 3.3.3 DEGES/`DESeq`).

3.3.1 DEGES/TbT

The DEGES/TbT pipeline for multi-group data is essentially the same as those for two-group data with/without replicates. Note that a smaller sampling size (i.e., `samplesize = 100`) is used here to reduce the computation time, but a larger sampling size of around 10,000 (i.e., `samplesize = 10000`) is recommended (Hardcastle and Kelly, 2010 [5]).

```
> set.seed(1000)
> library(TCC)
> data(hypoData_mg)
> samplesize <- 100
> group <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
> tcc <- new("TCC", hypoData_mg, group)
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "bayseq",
+                        iteration = 1, samplesize = samplesize)
> tcc$norm.factors
```

```
  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3   G3_rep1   G3_rep2
1.0369575 0.9165584 0.7861750 0.8229178 1.1755355 1.1965579 1.0224641 1.0227613
  G3_rep3
1.0200725
```

3.3.2 DEGES/edgeR

`edgeR` employs generalized linear models (GLMs) to find DEGs between any of the groups. The DEGES/`edgeR` normalization pipeline in `TCC` internally uses functions for the GLM approach that require two models (a full model and a null model). The full model corresponds to a design matrix to describe sample groups. The null model corresponds to the model coefficients. The two models can be defined as follows:

```
> library(TCC)
> data(hypoData_mg)
> group <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
> tcc <- new("TCC", hypoData_mg, group)
> design <- model.matrix(~ as.factor(group))
> coef <- 2:length(unique(group))
```

The design matrix (`design`) can be constructed by using the model.matrix function. For the model coefficients (`coef`), the user should specify all the coefficients except for the intercept term. The GLM-based DEGES/`edgeR` pipeline can be performed by adding the two arguments (`design` and `coef`) as follows:

```
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 1, design = design, coef = coef)
> tcc$norm.factors


  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3   G3_rep1   G3_rep2
1.0285193 0.9001479 0.7877150 0.8308582 1.1867099 1.1835535 0.9994667 1.0281489
  G3_rep3
1.0548806
```

For `edgeR` users, we provide commands, consisting of functions in `edgeR`, to perform the DEGES/`edgeR` pipeline without `TCC`. The `calcNormFators` function together with the above parameter settings can be regarded as a wrapper function for the following commands.

```
> library(TCC)
> data(hypoData_mg)
> group <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
> tcc <- new("TCC", hypoData_mg, group)
> FDR <- 0.1
> floorPDEG <- 0.05
> design <- model.matrix(~ as.factor(group))
> coef <- 2:length(unique(group))
> d <- DGEList(counts = hypoData_mg, group = group)
> ### STEP 1 ###
> d <- calcNormFactors(d)
> ### STEP 2 ###
> d <- estimateGLMCommonDisp(d, design)
> d <- estimateGLMTrendedDisp(d, design)
> d <- estimateGLMTagwiseDisp(d, design)
> fit <- glmFit(d, design)
> lrt <- glmLRT(fit, coef = coef)
> result <- topTags(lrt, n = nrow(hypoData_mg))
> result <- result$table[rownames(hypoData_mg), ]
> if (sum(result$FDR < FDR) > (floorPDEG * nrow(hypoData_mg))) {
+   is.DEG <- as.logical(result$FDR < FDR)
+ } else  {
+   is.DEG <- as.logical(rank(result$PValue, ties.method = "min") <=
+                        nrow(hypoData_mg) * floorPDEG)
+ }
```

```
> ### STEP 3 ###
> d <- DGEList(counts = hypoData_mg[!is.DEG, ], group = group)
> d <- calcNormFactors(d)
> norm.factors <- d$samples$norm.factors * colSums(hypoData_mg[!is.DEG, ]) /
+                 colSums(hypoData_mg)
> norm.factors <- norm.factors / mean(norm.factors)
> norm.factors
```

```
  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3   G3_rep1   G3_rep2
1.0285193 0.9001479 0.7877150 0.8308582 1.1867099 1.1835535 0.9994667 1.0281489
  G3_rep3
1.0548806
```

3.3.3 DEGES/DESeq

`DESeq` also employs GLMs for analyzing multi-group experiments. Similar to the `edgeR` package, it requires two models (full model and reduced model). The full model (`fit1`) and reduced model (`fit0`) can be created as follows:

```
> library(TCC)
> data(hypoData_mg)
> group <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
> tcc <- new("TCC", hypoData_mg, group)
> fit1 <- count ~ condition
> fit0 <- count ~ 1
```

The GLM-based DEGES/`DESeq` pipeline can be performed by adding the two arguments (`fit1` and `fit0`) as follows:

```
> tcc <- calcNormFactors(tcc, norm.method = "deseq", test.method = "deseq",
+                        iteration = 1, fit0 = fit0, fit1 = fit1)
> tcc$norm.factors
```

```
  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3   G3_rep1   G3_rep2
1.0232434 0.9096572 0.8017352 0.8249186 1.2025101 1.1728534 0.9884002 1.0164697
  G3_rep3
1.0602121
```

For `DESeq` users, we provide commands, consisting of functions in `DESeq`, to perform the DEGES/ `DESeq` pipeline without `TCC`. The `calcNormFators` function together with the above parameter settings can be regarded as a wrapper function for the following commands.

```
> library(TCC)
> data(hypoData_mg)
> group <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
> FDR <- 0.1
> floorPDEG <- 0.05
> tcc <- new("TCC", hypoData_mg, group)
> fit1 <- count ~ condition
```

```
> fit0 <- count ~ 1
> cds <- newCountDataSet(hypoData_mg, group)
> ### STEP 1 ###
> cds <- estimateSizeFactors(cds)
> ### STEP 2 ###
> cds <- estimateDispersions(cds)
> reduced.model <- fitNbinomGLMs(cds, fit0)

.

> full.model <- fitNbinomGLMs(cds, fit1)

.

> p.value <- nbinomGLMTest(full.model, reduced.model)
> p.value[is.na(p.value)] <- 1
> q.value <- p.adjust(p.value, method = "BH")
> if (sum(q.value < FDR) > (floorPDEG * nrow(hypoData_mg))) {
+   is.DEG <- as.logical(q.value < FDR)
+ } else {
+   is.DEG <- as.logical(rank(p.value, ties.method = "min") <=
+                        nrow(hypoData_mg) * floorPDEG)
+ }
> ### STEP 3 ###
> cds <- newCountDataSet(hypoData_mg[!is.DEG, ], group)
> cds <- estimateSizeFactors(cds)
> norm.factors <- sizeFactors(cds) / colSums(hypoData_mg)
> norm.factors <- norm.factors / mean(norm.factors)
> norm.factors

  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3   G3_rep1   G3_rep2
1.0232434 0.9096572 0.8017352 0.8249186 1.2025101 1.1728534 0.9884002 1.0164697
  G3_rep3
1.0602121
```

3.4 Retrieving normalized data

Similar functions for calculating normalization factors are the `calcNormFators` function in `edgeR` and the `estimateSizeFactors` function in `DESeq`. Note that the terminology used in `DESeq` (i.e., size factors) is different from that used in `edgeR` (i.e., effective library sizes) and ours. The effective library size in `edgeR` is calculated as the library size multiplied by the normalization factor. The size factors in the `DESeq` package are comparable to the *normalized* effective library sizes wherein the summary statistics for the effective library sizes are adjusted to one. Our normalization factors, which can be obtained from `tcc$norm.factors`, have the same names as those in `edgeR`. Accordingly, the normalization factors calculated from `TCC` with arbitrary options should be manipulated together with the library sizes when normalized read counts are to be

obtained. Since biologists are often interested in such information (Dillies et al., 2012 [3]), we provide the `getNormalizedData` function for retrieving normalized data.

Note that the `hypoData` consists of 1,000 genes and a total of six samples (three biological replicates for G1 and three biological replicates for G2); i.e., {G1_rep1, G1_rep2, G1_rep3} vs. {G2_rep1, G2_rep2, G2_rep3}. These simulation data have basically the same conditions as shown in Fig. 1 of the TbT paper (Kadota et al., 2012 [6]); i.e., (i) the first 200 genes are DEGs ($P_{\mathrm{DEG}} = 200/1000 = 20\%$), (ii) the first 180 genes of the 200 DEGs are higher in G1 ($P_{\mathrm{G1}} = 180/200 = 90\%$), and the remaining 20 DEGs are higher in G2, and (iii) the level of DE is four-fold. The last 800 genes were designed to be non-DEGs. The different normalization strategies can roughly be evaluated in terms of the similarity of their summary statistics for *normalized* data labeled as non-DEGs in one group (e.g., G1) to those of the other group (e.g., G2). The basic statistics for the non-DEGs are as follows.

```
> library(TCC)
> data(hypoData)
> nonDEG <- 201:1000
> summary(hypoData[nonDEG, ])
```

```
    G1_rep1           G1_rep2        G1_rep3           G2_rep1
 Min.   :   0.00   Min.   :   0   Min.   :   0.00   Min.   :   0.0
 1st Qu.:   3.00   1st Qu.:   4   1st Qu.:   3.00   1st Qu.:   3.0
 Median :  20.50   Median :  20   Median :  20.00   Median :  21.0
 Mean   : 103.36   Mean   : 105   Mean   : 104.45   Mean   : 113.8
 3rd Qu.:  74.25   3rd Qu.:  68   3rd Qu.:  73.25   3rd Qu.:  68.0
 Max.   :8815.00   Max.   :9548   Max.   :8810.00   Max.   :9304.0
    G2_rep2        G2_rep3
 Min.   :   0   Min.   :   0.0
 1st Qu.:   3   1st Qu.:   3.0
 Median :  21   Median :  20.0
 Mean   : 105   Mean   : 104.6
 3rd Qu.:  70   3rd Qu.:  70.0
 Max.   :9466   Max.   :9320.0
```

From now on, we will display only the median values for simplicity, i.e.,

```
> apply(hypoData[nonDEG, ], 2, median)
```

```
G1_rep1 G1_rep2 G1_rep3 G2_rep1 G2_rep2 G2_rep3
   20.5    20.0    20.0    21.0    21.0    20.0
```

In what follows, we show detailed examples using `hypoData`. Note, however, that the basic usage is simple.

```
> normalized.count <- getNormalizedData(tcc)
```

3.4.1 Retrieving two-group DEGES/edgeR-normalized data with replicates

The `getNormalizedData` function can be applied to the `TCC` class object after the normalization factors have been calculated.

```
> library(TCC)
> data(hypoData)
> nonDEG <- 201:1000
> group <- c(1, 1, 1, 2, 2, 2)
> tcc <- new("TCC", hypoData, group)
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 1, FDR = 0.1, floorPDEG = 0.05)
> normalized.count <- getNormalizedData(tcc)
> apply(normalized.count[nonDEG, ], 2, median)


 G1_rep1  G1_rep2  G1_rep3  G2_rep1  G2_rep2  G2_rep3
20.26002 20.01902 19.50410 21.30174 21.52711 19.95349
```

The same procedure consisting of functions in `edgeR` is

```
> library(TCC)
> data(hypoData)
> nonDEG <- 201:1000
> group <- c(1, 1, 1, 2, 2, 2)
> FDR <- 0.1
> floorPDEG <- 0.05
> d <- DGEList(counts = hypoData, group = group)
> ###  Step 1  ###
> d <- calcNormFactors(d)
> ###  Step 2  ###
> d <- estimateCommonDisp(d)
> d <- estimateTagwiseDisp(d)
> result <- exactTest(d)
> q.value <- p.adjust(result$table$PValue, method = "BH")
> if (sum(q.value < FDR) > (floorPDEG * nrow(hypoData))) {
+   is.DEG <- as.logical(q.value < FDR)
+ } else {
+   is.DEG <- as.logical(order(rank(result$table$PValue)) <=
+                        nrow(hypoData) * floorPDEG)
+ }
> ###  Step 3  ###
> d <- DGEList(counts = hypoData[!is.DEG, ], group = group)
> d <- calcNormFactors(d)
> norm.factors <- d$samples$norm.factors * colSums(hypoData[!is.DEG, ]) /
+                   colSums(hypoData)
> norm.factors <- norm.factors / mean(norm.factors)
> effective.libsizes <- colSums(hypoData) * norm.factors
> normalized.count <- sweep(hypoData, 2,
+                           mean(effective.libsizes) / effective.libsizes, "*")
> apply(normalized.count[nonDEG, ], 2, median)
```

```
 G1_rep1  G1_rep2  G1_rep3  G2_rep1  G2_rep2  G2_rep3
20.26002 20.01902 19.50410 21.30174 21.52711 19.95349
```

It is obvious that the summary statistics (ranging from 19.50410 to 21.52711) from DEGES/edgeR-normalized data are close to the truth (i.e., ranging from 20.0 to 21.0). For comparison, the summary statistics for TMM-normalized data produced using the original normalization method (i.e., TMM) in edgeR are obtained as follows.

```
> library(TCC)
> data(hypoData)
> nonDEG <- 201:1000
> group <- c(1, 1, 1, 2, 2, 2)
> d <- DGEList(count = hypoData, group = group)
> d <- calcNormFactors(d)
> norm.factors <- d$samples$norm.factors
> norm.factors <- norm.factors / mean(norm.factors)
> effective.libsizes <- colSums(hypoData) * norm.factors
> normalized.count <- sweep(hypoData, 2,
+                           mean(effective.libsizes) / effective.libsizes, "*")
> apply(normalized.count[nonDEG, ], 2, median)


 G1_rep1  G1_rep2  G1_rep3  G2_rep1  G2_rep2  G2_rep3
19.35893 19.01078 18.59060 22.98591 22.16273 21.00685
```

This is the same as

```
> library(TCC)
> data(hypoData)
> nonDEG <- 201:1000
> group <- c(1, 1, 1, 2, 2, 2)
> tcc <- new("TCC", hypoData, group)
> tcc <- calcNormFactors(tcc, norm.method = "tmm", iteration = 0)
> normalized.count <- getNormalizedData(tcc)
> apply(normalized.count[nonDEG, ], 2, median)


 G1_rep1  G1_rep2  G1_rep3  G2_rep1  G2_rep2  G2_rep3
19.35893 19.01078 18.59060 22.98591 22.16273 21.00685
```

From the viewpoint of the data distribution of non-DEGs, these statistics (ranging from 18.59060 to 22.98591) are not as good as those of DEGES/edgeR.

3.4.2 Retrieving two-group DEGES/DESeq-normalized data with replicates

Similar to the DEGES/edgeR case, DEGES/DESeq-normalized data can be retrieved as follows.

```
> library(TCC)
> data(hypoData)
> group <- c(1, 1, 1, 2, 2, 2)
> nonDEG <- 201:1000
> tcc <- new("TCC", hypoData, group)
```

```
> tcc <- calcNormFactors(tcc, norm.method = "deseq", test.method = "deseq",
+                        iteration = 1, FDR = 0.1, floorPDEG = 0.05)
> normalized.count <- getNormalizedData(tcc)
> apply(normalized.count[nonDEG, ], 2, median)


 G1_rep1  G1_rep2  G1_rep3  G2_rep1  G2_rep2  G2_rep3
19.98051 19.23724 19.79865 21.56052 21.57241 20.47685
```

The same procedure consisting of functions in `DESeq` is

```
> library(TCC)
> data(hypoData)
> nonDEG <- 201:1000
> group <- c(1, 1, 1, 2, 2, 2)
> FDR <- 0.1
> floorPDEG <- 0.05
> cds <- newCountDataSet(hypoData, group)
> ###  Step 1  ###
> cds <- estimateSizeFactors(cds)
> ###  Step 2  ###
> cds <- estimateDispersions(cds)
> result <- nbinomTest(cds, 1, 2)
> result$pval[is.na(result$pval)] <- 1
> result$padj[is.na(result$padj)] <- 1
> q.value <- result$padj
> if (sum(q.value < FDR) > (floorPDEG * nrow(hypoData))) {
+   is.DEG <- as.logical(q.value < FDR)
+ } else {
+   is.DEG <- as.logical(rank(result$pval, ties.method = "min") <=
+                        nrow(hypoData) * floorPDEG)
+ }
> ###  Step 3  ###
> cds <- newCountDataSet(hypoData[!is.DEG, ], group)
> cds <- estimateSizeFactors(cds)
> norm.factors <- sizeFactors(cds) / colSums(hypoData)
> norm.factors <- norm.factors / mean(norm.factors)
> effective.libsizes <- colSums(hypoData) * norm.factors
> normalized.count <- sweep(hypoData, 2,
+                           mean(effective.libsizes) / effective.libsizes, "*")
> apply(normalized.count[nonDEG, ], 2, median)


 G1_rep1  G1_rep2  G1_rep3  G2_rep1  G2_rep2  G2_rep3
19.98051 19.23724 19.79865 21.56052 21.57241 20.47685
```

3.4.3 Retrieving two-group DEGES/DESeq-normalized data without replicates

Similar to the case of count data with replicates, the DEGES/`DESeq`-normalized data without replicates can be retrieved as follows.

```
> library(TCC)
> data(hypoData)
> nonDEG <- 201:1000
> group <- c(1, 2)
> tcc <- new("TCC", hypoData[, c(1, 4)], group)
> tcc <- calcNormFactors(tcc, norm.method = "deseq", test.method = "deseq",
+                        iteration = 1, FDR = 0.1, floorPDEG = 0.05)
> normalized.count <- getNormalizedData(tcc)
> apply(normalized.count[nonDEG, ], 2, median)


 G1_rep1  G2_rep1
19.70555 21.88220
```

The same procedure consisting of functions in `DESeq` is

```
> library(TCC)
> data(hypoData)
> nonDEG <- 201:1000
> group <- c(1, 2)
> FDR <- 0.1
> floorPDEG <- 0.05
> cds <- newCountDataSet(hypoData[,c(1, 4)], group)
> ###  Step 1  ###
> cds <- estimateSizeFactors(cds)
> ###  Step 2  ###
> cds <- estimateDispersions(cds, method = "blind", sharingMode = "fit-only")
> result <- nbinomTest(cds, 1, 2)
> result$pval[is.na(result$pval)] <- 1
> result$padj[is.na(result$padj)] <- 1
> q.value <- result$padj
> if (sum(q.value < FDR) > (floorPDEG * nrow(hypoData))) {
+   is.DEG <- as.logical(q.value < FDR)
+ } else {
+   is.DEG <- as.logical(rank(result$pval, ties.method = "min") <=
+                        nrow(hypoData) * floorPDEG)
+ }
> ###  Step 3  ###
> cds <- newCountDataSet(hypoData[!is.DEG, c(1, 4)], group)
> cds <- estimateSizeFactors(cds)
> norm.factors <- sizeFactors(cds) / colSums(hypoData[, c(1, 4)])
> norm.factors <- norm.factors / mean(norm.factors)
> effective.libsizes <- colSums(hypoData[, c(1, 4)]) * norm.factors
> normalized.count <- sweep(hypoData[, c(1, 4)], 2,
+                           mean(effective.libsizes) / effective.libsizes, "*")
> apply(normalized.count[nonDEG, ], 2, median)


 G1_rep1  G2_rep1
19.70555 21.88220
```

The above summary statistics from DEGES/DESeq-normalized data are closer to the truth (i.e., 20.5 for G1_rep1 and 21.0 for G2_rep1) than are the following summary statistics from data normalized using the original normalization method implemented in DESeq.

```
> library(TCC)
> data(hypoData)
> nonDEG <- 201:1000
> group <- c(1, 2)
> cds <- newCountDataSet(hypoData[, c(1, 4)], group)
> cds <- estimateSizeFactors(cds)
> normalized.count <- counts(cds, normalized = TRUE)
> apply(normalized.count[nonDEG, ], 2, median)

 G1_rep1  G2_rep1
19.40717 22.18253
```

3.4.4 Retrieving multi-group iDEGES/edgeR-normalized data with replicates

Here, we analyze another hypothetical three-group count matrix, the hypoData_mg object, provided in TCC. It consists of 1,000 genes and a total of nine columns for testing any difference among three groups that each have triplicates. Similar to the hypoData object, the first 200 genes are DEGs and the remaining 800 genes are non-DEGs. The basic statistics for the non-DEGs are as follows.

```
> library(TCC)
> data(hypoData_mg)
> nonDEG <- 201:1000
> summary(hypoData_mg[nonDEG, ])

    G1_rep1           G1_rep2           G1_rep3           G2_rep1
 Min.   :    0.00   Min.   :    0.0   Min.   :    0.0   Min.   :    0.0
 1st Qu.:    2.00   1st Qu.:    2.0   1st Qu.:    2.0   1st Qu.:    2.0
 Median :   14.00   Median :   13.0   Median :   14.5   Median :   13.0
 Mean   :  135.41   Mean   :  150.5   Mean   :  190.6   Mean   :  199.4
 3rd Qu.:   51.25   3rd Qu.:   53.0   3rd Qu.:   55.0   3rd Qu.:   52.0
 Max.   :27218.00   Max.   :27987.0   Max.   :66273.0   Max.   :75148.0
    G2_rep2           G2_rep3           G3_rep1          G3_rep2
 Min.   :    0.00   Min.   :    0.0   Min.   :    0   Min.   :    0.0
 1st Qu.:    2.00   1st Qu.:    2.0   1st Qu.:    2   1st Qu.:    2.0
 Median :   13.00   Median :   14.0   Median :   14   Median :   15.0
 Mean   :  132.53   Mean   :  138.4   Mean   :  164   Mean   :  166.2
 3rd Qu.:   52.25   3rd Qu.:   55.0   3rd Qu.:   52   3rd Qu.:   55.0
 Max.   :22381.00   Max.   :24979.0   Max.   :49398   Max.   :49709.0
    G3_rep3
 Min.   :    0.0
 1st Qu.:    2.0
 Median :   15.0
 Mean   :  152.1
 3rd Qu.:   50.0
 Max.   :39299.0
```

From now on, we will display only the median values for simplicity, i.e.,

```
> apply(hypoData_mg[nonDEG, ], 2, median)
```

```
G1_rep1 G1_rep2 G1_rep3 G2_rep1 G2_rep2 G2_rep3 G3_rep1 G3_rep2 G3_rep3
   14.0    13.0    14.5    13.0    13.0    14.0    14.0    15.0    15.0
```

The iDEGES/edgeR-normalized data can be retrieved as follows.

```
> library(TCC)
> data(hypoData_mg)
> nonDEG <- 201:1000
> group <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
> tcc <- new("TCC", hypoData_mg, group)
> design <- model.matrix(~ as.factor(group))
> coef <- 2:length(unique(group))
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 3, design = design, coef = coef)
> normalized.count <- getNormalizedData(tcc)
> apply(normalized.count[nonDEG, ], 2, median)
```

```
 G1_rep1  G1_rep2  G1_rep3  G2_rep1  G2_rep2  G2_rep3  G3_rep1  G3_rep2
13.86333 13.21511 14.07631 13.00057 13.31609 13.73010 14.24628 14.74515
 G3_rep3
15.29914
```

```
> range(apply(normalized.count[nonDEG, ], 2, median))
```

```
[1] 13.00057 15.29914
```

For comparison, the summary statistics for TMM-normalized data produced using the original normalization method (i.e., TMM) in edgeR are obtained as follows.

```
> library(TCC)
> data(hypoData_mg)
> nonDEG <- 201:1000
> group <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
> tcc <- new("TCC", hypoData_mg, group)
> tcc <- calcNormFactors(tcc, norm.method = "tmm", iteration = 0)
> normalized.count <- getNormalizedData(tcc)
> apply(normalized.count[nonDEG, ], 2, median)
```

```
 G1_rep1  G1_rep2  G1_rep3  G2_rep1  G2_rep2  G2_rep3  G3_rep1  G3_rep2
13.46685 12.55871 13.42509 13.14068 13.52699 13.84316 14.20297 15.38684
 G3_rep3
16.25185
```

```
> range(apply(normalized.count[nonDEG, ], 2, median))
```

```
[1] 12.55871 16.25185
```

It is obvious that the summary statistics (ranging from 13.00057 to 15.29914) from iDEGES/`edgeR`-normalized data are closer to the truth (i.e., ranging from 13.0 to 15.0) than those (ranging from 12.55871 to 16.25185) from TMM-normalized data.

4 Differential expression (DE)

The particular feature of TCC is that it calculates robust normalization factors. Moreover, end users would like to have some accessory functions for subsequent analyses. Here, we provide the estimateDE function for identifying DEGs. Specifically, the function internally uses the corresponding functions implemented in three packages: exactTest in edgeR, nbinomTest in DESeq, and getLikelihoods.NB in baySeq. Similar to the usage in the calcNormFators function with the test.method argument in TCC, those DE methods in edgeR, DESeq, and baySeq can be performed by using the estimateDE function with test.method = "edger", "deseq", and "bayseq", respectively. Here, we show some examples of DE analysis for two-group data with replicates (4.1), two-group data without replicates (4.2), and multi-group data with replicates (4.3).

4.1 DE analysis for two-group data with replicates

4.1.1 edgeR coupled with iDEGES/edgeR normalization

We give a procedure for DE analysis using the exact test implemented in edgeR together with iDEGES/edgeR normalization factors (i.e., the iDEGES/edgeR-edgeR combination) for the hypothetical two-group count data with replicates (i.e., the hypoData object). If the user wants to determine the genes having an FDR threshold of $< 10\%$ as DEGs, one can do as follows.

```
> library(TCC)
> data(hypoData)
> group <- c(1, 1, 1, 2, 2, 2)
> tcc <- new("TCC", hypoData, group)
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 3, FDR = 0.1, floorPDEG = 0.05)
> tcc <- estimateDE(tcc, test.method = "edger", FDR = 0.1)
```

The results of the DE analysis are stored in the TCC class object. The summary statistics for top-ranked genes can be retrieved by using the getResult function.

```
> result <- getResult(tcc, sort = TRUE)
> head(result)
```

```
     gene_id  a.value   m.value      p.value      q.value rank estimatedDEG
151 gene_151 9.736785 -2.753816 4.641083e-11 4.641083e-08    1            1
39   gene_39 7.110842 -2.460691 9.115042e-10 3.270783e-07    2            1
599 gene_599 5.927173 -3.282264 9.812348e-10 3.270783e-07    3            1
68   gene_68 6.209395 -2.867694 4.776945e-09 1.194236e-06    4            1
175 gene_175 7.984265 -2.373657 1.082493e-08 1.899827e-06    5            1
144 gene_144 7.588164 -2.130092 1.139896e-08 1.899827e-06    6            1
```

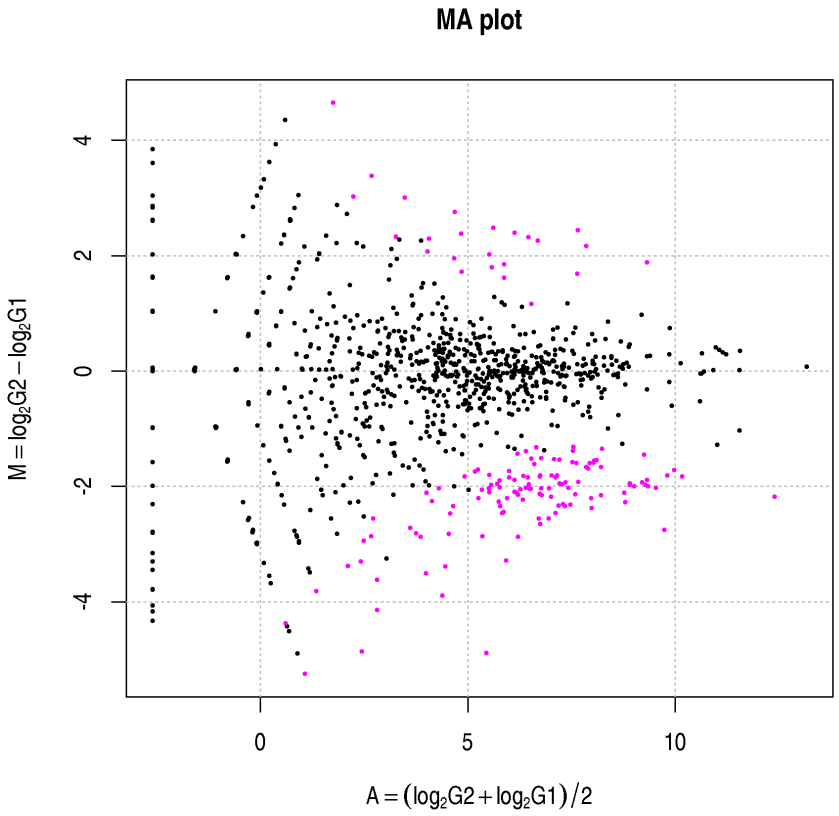
The DE results can be broken down as follows.

```
> table(tcc$estimatedDEG)
```

```
  0   1
854 146
```

This means 854 non-DEGs and 146 DEGs satisfy FDR < 0.1. The `plot` function generates an M-A plot, where "M" indicates the log-ratio (i.e., M $= log_2$G2$-log_2$G1) and "A" indicates average read count (i.e., A $= (log_2$G2$+log_2$G1$)/2$), from the normalized count data. The magenta points indicate the identified DEGs at FDR < 0.1.

```
> plot(tcc)
```



4.1.2 baySeq coupled with iDEGES/edgeR normalization

If the user wants to employ the empirical Bayesian method in `baySeq` together with iDEGES/`edgeR` normalization factors (i.e., the iDEGES/`edgeR`-`baySeq` combination), one can do as follows.

```
> set.seed(1000)
> library(TCC)
> data(hypoData)
> samplesize <- 100
> group <- c(1, 1, 1, 2, 2, 2)
> tcc <- new("TCC", hypoData, group)
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 3, FDR = 0.1, floorPDEG = 0.05)
> tcc <- estimateDE(tcc, test.method = "bayseq",
```

```
+                   FDR = 0.1, samplesize = samplesize)
> result <- getResult(tcc, sort = TRUE)
> head(result)

     gene_id  a.value   m.value      p.value      q.value rank estimatedDEG
168 gene_168 8.903341 -1.968787 9.181013e-08 9.181013e-08    1            1
115 gene_115 8.903382 -1.947266 2.164326e-07 1.541213e-07    2            1
171 gene_171 8.795243 -2.267941 1.240757e-05 4.238604e-06    3            1
176 gene_176 9.013616 -1.999303 7.287976e-05 2.139889e-05    4            1
144 gene_144 7.588164 -2.130092 1.230200e-04 4.172310e-05    5            1
3     gene_3 9.251797 -1.446596 1.435138e-04 5.868823e-05    6            1

> table(tcc$estimatedDEG)

  0   1
869 131
```

Note that a smaller sampling size (i.e., `samplesize = 100`) is used here to reduce the computation time, but a larger sampling size of around $10,000$ (i.e., `samplesize = 10000`) is recommended (Hardcastle and Kelly, 2010 [5]). Note also that `baySeq` outputs posterior likelihoods instead of the p-values obtained from `edgeR` and `DESeq`. The p-value column stores the $(1 - likelihood)$ values when the `estimateDE` function is executed with the empirical Bayes in `baySeq`. Now let us describe an alternative procedure for `baySeq` users that corresponds to the `estimateDE` function. The *likelihood* values and p-values (calculated as $1 - likelihood$) are retrieved as follows.

```
> set.seed(1000)
> library(TCC)
> data(hypoData)
> samplesize <- 100
> group <- c(1, 1, 1, 2, 2, 2)
> tcc <- new("TCC", hypoData, group)
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 3, FDR = 0.1, floorPDEG = 0.05)
> effective.libsizes <- colSums(tcc$count) * tcc$norm.factors
> groups <- list(NDE = rep(1, length(group)), DE = group)
> cD <- new("countData", data = tcc$count, replicates = group,
+           libsizes = effective.libsizes, groups = groups)
> cD <- getPriors.NB(cD, samplesize = samplesize,
+                    estimation = "QL", cl = NULL)
> cD <- getLikelihoods.NB(cD, pET = "BIC", cl = NULL)

.

> tmp <- topCounts(cD, group = "DE", number = nrow(tcc$count))
> p.value <- 1 - tmp$Likelihood
> q.value <- tmp$FDR
> result <- cbind(p.value, q.value)
> rownames(result) <- rownames(tmp)
> head(result)
```

```
             p.value      q.value
gene_168 9.181013e-08 9.181013e-08
gene_115 2.164326e-07 1.541213e-07
gene_171 1.240757e-05 4.238604e-06
gene_176 7.287976e-05 2.139889e-05
gene_144 1.230200e-04 4.172310e-05
gene_3   1.435138e-04 5.868823e-05
```

4.2 DE analysis for two-group data without replicates

As described previously, the functions for the DEG identification method implemented in `edgeR` (i.e., the exact test; ver. 3.0.4) do not allow analysis without replicates. Currently, the `estimateDE` function only allows the `"deseq"` or `"bayseq"` options for the `test.method` argument. Here, we show a procedure for DE analysis using the NB test implemented in `DESeq` together with iDEGES/`DESeq` normalization factors (i.e., the iDEGES/`DESeq`-`DESeq` combination) for the hypothetical two-group count data without replicates (i.e., the `hypoData[, c(1, 4)]` object). If the user wants to determine the genes having an FDR threshold of $< 10\%$ as DEGs, one can do as follows.

```
> library(TCC)
> data(hypoData)
> group <- c(1, 2)
> tcc <- new("TCC", hypoData[, c(1, 4)], group)
> head(tcc$count)
```

```
       G1_rep1 G2_rep1
gene_1      34      16
gene_2     358      36
gene_3    1144     374
gene_4       0      18
gene_5      98       1
gene_6     296      86
```

```
> tcc$group
```

```
        group
G1_rep1     1
G2_rep1     2
```

```
> tcc <- calcNormFactors(tcc, norm.method = "deseq", test.method = "deseq",
+                        iteration = 3, FDR = 0.1, floorPDEG = 0.05)
> tcc$norm.factors
```

```
  G1_rep1   G2_rep1
0.9211464 1.0788536
```

```
> tcc <- estimateDE(tcc, test.method = "deseq",
+                   FDR = 0.1)
> result <- getResult(tcc, sort = TRUE)
> head(result)
```

```
     gene_id    a.value   m.value      p.value   q.value rank estimatedDEG
36   gene_36 -0.9988563 -8.525340 0.0002119999 0.2011879    1            0
17   gene_17  5.9635499 -5.470058 0.0064590145 1.0000000    2            0
5     gene_5  3.3084986 -6.499805 0.0184836363 1.0000000    3            0
989 gene_989 -0.9988563 -5.839291 0.0252092271 1.0000000    4            0
187 gene_187  3.6973024  5.507222 0.0321082546 1.0000000    5            0
822 gene_822 -0.9988563  5.436833 0.0522752310 1.0000000    6            0
```

```
> table(tcc$estimatedDEG)
```

```
   0
1000
```

It can be seen that there is no DEG having FDR < 0.1.

4.3 DE analysis for multi-group data with replicates

Here, we give three examples of DE analysis coupled with DEGES/`edgeR` normalization for the hypothetical three-group data with replicates, i.e., the `hypoData_mg` object. The use of the DEGES/`edgeR` normalization factors is simply for reducing the computation time.

4.3.1 baySeq coupled with DEGES/edgeR normalization

The empirical Bayesian method implemented in `baySeq` after executing the DEGES/`edgeR` normalization (i.e., the DEGES/`edgeR`-`baySeq` combination) can be performed as follows.

```
> library(TCC)
> data(hypoData_mg)
> group <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
> tcc <- new("TCC", hypoData_mg, group)
> ###  Normalization  ###
> design <- model.matrix(~ as.factor(group))
> coef <- 2:length(unique(group))
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 1, design = design, coef = coef)
> ###  DE analysis  ###
> set.seed(1000)
> samplesize <- 100
> tcc <- estimateDE(tcc, test.method = "bayseq",
+                   FDR = 0.1, samplesize = samplesize)
> result <- getResult(tcc, sort = TRUE)
> head(result)
```

```
     gene_id a.value m.value      p.value      q.value rank estimatedDEG
27   gene_27      NA      NA 9.612982e-08 9.612982e-08    1            1
134 gene_134      NA      NA 1.706725e-07 1.334012e-07    2            1
194 gene_194      NA      NA 3.685956e-07 2.117993e-07    3            1
179 gene_179      NA      NA 4.687318e-07 2.760324e-07    4            1
74   gene_74      NA      NA 2.131801e-06 6.471862e-07    5            1
169 gene_169      NA      NA 2.939300e-06 1.029205e-06    6            1
```

```
> table(tcc$estimatedDEG)
```

```
  0   1
897 103
```

It can be seen that the `baySeq` method identified 103 DEGs having FDR < 0.1. One can obtain the number of DEGs with another threshold (e.g., FDR < 0.2) from the result object as follows.

```
> sum(result$q.value < 0.2)
```

```
[1] 131
```

For `baySeq` users, we provide commands, consisting of functions in `baySeq`, to perform the DEG identification without the function in `TCC`. The `estimateDE` function with `test.method = "bayseq"` can be regarded as a wrapper function for the following commands after the DEGES/`edgeR` normalization.

```
> set.seed(1000)
> samplesize <- 100
> effective.libsizes <- colSums(tcc$count) * tcc$norm.factors
> groups <- list(NDE = rep(1, length(group)), DE = group)
> cD <- new("countData", data = tcc$count, replicates = group,
+           libsizes = effective.libsizes, groups = groups)
> cD <- getPriors.NB(cD, samplesize = samplesize,
+                    estimation = "QL", cl = NULL)
> cD <- getLikelihoods.NB(cD, pET = "BIC", cl = NULL)
```

```
.
```

```
> tmp <- topCounts(cD, group = "DE", number = nrow(tcc$count))
> p.value <- 1 - tmp$Likelihood
> q.value <- tmp$FDR
> result <- cbind(p.value, q.value)
> rownames(result) <- rownames(tmp)
> head(result)
```

```
              p.value      q.value
gene_27  9.612982e-08 9.612982e-08
gene_134 1.706725e-07 1.334012e-07
gene_194 3.685956e-07 2.117993e-07
gene_179 4.687318e-07 2.760324e-07
gene_74  2.131801e-06 6.471862e-07
gene_169 2.939300e-06 1.029205e-06
```

```
> sum(q.value < 0.1)
```

```
[1] 103
```

```
> sum(q.value < 0.2)
```

```
[1] 131
```

4.3.2 edgeR coupled with DEGES/edgeR normalization

The exact test implemented in `edgeR` after executing the DEGES/`edgeR` normalization (i.e., the DEGES/`edgeR`-`edgeR` combination) can be performed as follows.

```
> library(TCC)
> data(hypoData_mg)
> group <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
> tcc <- new("TCC", hypoData_mg, group)
> ###  Normalization  ###
> design <- model.matrix(~ as.factor(group))
> coef <- 2:length(unique(group))
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 1, design = design, coef = coef)
> ###  DE analysis  ###
> tcc <- estimateDE(tcc, test.method = "edger",
+                   FDR = 0.1, design = design, coef = coef)
> result <- getResult(tcc, sort = TRUE)
> head(result)
```

```
     gene_id a.value m.value      p.value      q.value rank estimatedDEG
176 gene_176      NA      NA 2.207155e-13 2.207155e-10    1            1
27   gene_27      NA      NA 4.924858e-12 2.004363e-09    2            1
126 gene_126      NA      NA 6.013088e-12 2.004363e-09    3            1
56   gene_56      NA      NA 8.316457e-12 2.079114e-09    4            1
64   gene_64      NA      NA 2.596476e-11 4.369393e-09    5            1
121 gene_121      NA      NA 2.621636e-11 4.369393e-09    6            1
```

```
> table(tcc$estimatedDEG)
```

```
  0   1
828 172
```

Note that these DEGs having FDR < 0.1 display DE between any of the groups because the two arguments indicated here (`design` and `coef`) correspond to an AVOVA-like test for any differences provided in `edgeR`, i.e.,

```
> library(TCC)
> data(hypoData_mg)
> group <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
> tcc <- new("TCC", hypoData_mg, group)
> ###  Normalization  ###
> design <- model.matrix(~ as.factor(group))
> coef <- 2:length(unique(group))
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 1, design = design, coef = coef)
> ###  DE analysis  ###
> d <- DGEList(tcc$count, group = group)
> d$samples$norm.factors <- tcc$norm.factors
> d <- estimateGLMCommonDisp(d, design)
> d <- estimateGLMTrendedDisp(d, design)
> d <- estimateGLMTagwiseDisp(d, design)
> fit <- glmFit(d, design)
> lrt <- glmLRT(fit, coef = coef)
> tmp <- topTags(lrt, n = nrow(tcc$count))
> p.value <- tmp$table$PValue
> q.value <- tmp$table$FDR
> result <- cbind(p.value, q.value)
> rownames(result) <- rownames(tmp)
> head(result)
```

```
              p.value      q.value
gene_176 2.207155e-13 2.207155e-10
gene_27  4.924858e-12 2.004363e-09
gene_126 6.013088e-12 2.004363e-09
gene_56  8.316457e-12 2.079114e-09
gene_64  2.596476e-11 4.369393e-09
gene_121 2.621636e-11 4.369393e-09
```

```
> sum(q.value < 0.1)
```

```
[1] 172
```

```
> sum(q.value < 0.2)
```

```
[1] 209
```

As described in the edgeR manual, the second and third columns in the design object are relative to the baseline (i.e., Group 1 or G1): coef = 2 means G2 vs. G1 and coef = 3 means G3 vs. G1. The above procedure with the coef object (i.e., 2:length(unique(group))) indicates the both comparisons (i.e., G2 vs. G1 and G3 vs. G1) and identifies DEGs between any of the three groups. In other words, one can do any two-group comparison of interest from multi-group data with replicates. For example, the DE analysis for G3 vs. G1 together with DEGES/edgeR normalization can be performed as follows.

```
> library(TCC)
> data(hypoData_mg)
> group <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
> tcc <- new("TCC", hypoData_mg, group)
> ###  Normalization  ###
> design <- model.matrix(~ as.factor(group))
> coef <- 2:length(unique(group))
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 1, design = design, coef = coef)
> ###  DE analysis  ###
> coef <- 3
> tcc <- estimateDE(tcc, test.method = "edger",
+                   FDR = 0.1, design = design, coef = coef)
> result <- getResult(tcc, sort = TRUE)
> head(result)
```

```
     gene_id a.value m.value      p.value      q.value rank estimatedDEG
126 gene_126      NA      NA 8.793286e-10 8.793286e-07    1            1
56   gene_56      NA      NA 6.442465e-09 2.613289e-06    2            1
27   gene_27      NA      NA 7.839868e-09 2.613289e-06    3            1
186 gene_186      NA      NA 2.085647e-08 4.301871e-06    4            1
121 gene_121      NA      NA 2.150935e-08 4.301871e-06    5            1
28   gene_28      NA      NA 3.175588e-08 4.920537e-06    6            1
```

```
> table(tcc$estimatedDEG)
```

```
  0   1
881 119
```

4.3.3 DESeq coupled with DEGES/edgeR normalization

The NB test implemented in DESeq after executing the DEGES/edgeR normalization (i.e., the DEGES/edgeR-DESeq combination) can be performed as follows.

```
> library(TCC)
> data(hypoData_mg)
> group <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
> tcc <- new("TCC", hypoData_mg, group)
> ###  Normalization  ###
```

```
> design <- model.matrix(~ as.factor(group))
> coef <- 2:length(unique(group))
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 1, design = design, coef = coef)
> ###  DE analysis  ###
> fit1 <- count ~ condition
> fit0 <- count ~ 1
> tcc <- estimateDE(tcc, test.method = "deseq",
+                   FDR = 0.1, fit0 = fit0, fit1 = fit1)
> result <- getResult(tcc, sort = TRUE)
> head(result)
```

```
     gene_id a.value m.value      p.value      q.value rank estimatedDEG
126 gene_126      NA      NA 2.182698e-13 2.182698e-10    1            1
63   gene_63      NA      NA 1.244090e-10 6.220452e-08    2            1
27   gene_27      NA      NA 2.134407e-08 7.114689e-06    3            1
176 gene_176      NA      NA 4.139815e-08 1.034954e-05    4            1
83   gene_83      NA      NA 5.531305e-08 1.106261e-05    5            1
121 gene_121      NA      NA 6.736260e-08 1.122710e-05    6            1
```

```
> table(tcc$estimatedDEG)
```

```
  0   1
870 130
```

For `DESeq` users, we provide commands, consisting of functions in `DESeq`, to perform the DEG identification without the function in `TCC`. The `estimateDE` function with `test.method = "deseq"` can be regarded as a wrapper function for the following commands after the DEGES/`edgeR` normalization.

```
> library(TCC)
> data(hypoData_mg)
> group <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
> tcc <- new("TCC", hypoData_mg, group)
> ###  Normalization  ###
> design <- model.matrix(~ as.factor(group))
> coef <- 2:length(unique(group))
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 1, design = design, coef = coef)
> ###  DE analysis  ###
> fit1 <- count ~ condition
> fit0 <- count ~ 1
> cds <- newCountDataSet(tcc$count, group)
> sizeFactors(cds) <- tcc$norm.factors * colSums(tcc$count)
> cds <- estimateDispersions(cds)
> reduced.model <- fitNbinomGLMs(cds, fit0)
```

.

```
> full.model <- fitNbinomGLMs(cds, fit1)

.

> p.value <- nbinomGLMTest(full.model, reduced.model)
> p.value[is.na(p.value)] <- 1
> q.value <- p.adjust(p.value, method = "BH")
> tmp <- cbind(p.value, q.value)
> rownames(tmp) <- tcc$gene_id
> result <- tmp[order(p.value), ]
> head(result)

              p.value      q.value
gene_126 2.182698e-13 2.182698e-10
gene_63  1.244090e-10 6.220452e-08
gene_27  2.134407e-08 7.114689e-06
gene_176 4.139815e-08 1.034954e-05
gene_83  5.531305e-08 1.106261e-05
gene_121 6.736260e-08 1.122710e-05

> sum(q.value < 0.1)

[1] 130

> sum(q.value < 0.2)

[1] 143
```

5 Generation of simulation data

5.1 Introduction and basic usage

As demonstrated in our previous study (Kadota et al., 2012 [6]), the DEGES-based normalization methods implemented in `TCC` theoretically outperform the other normalization methods when the numbers of DEGs (G1 vs. G2) in the tag count data are biased. However, it is difficult to determine whether the up- and down-regulated DEGs in one of the groups are actually biased in their number when analyzing real data (Dillies et al., 2012 [3]). This means we have to evaluate the potential performance of our DEGES-based methods using mainly simulation data. The `simulateReadCounts` function generates simulation data under various conditions. This function can generate simulation data analyzed in the TbT paper (Kadota et al., 2012 [6]), and that means it enables other researchers to compare the methods they develop with our DEGES-based methods. For example, the `hypoData` object, a hypothetical count dataset provided in `TCC`, was generated by using this function. The output of the `simulateReadCounts` function is stored as a `TCC` class object and is therefore ready-to-analyze.

Note that different trials of simulation analysis generally yield different count data even under the same simulation conditions. As mentioned in section 3.1.1, we can call the `set.seed` function in order to obtain reproducible results (i.e., the `tcc$count`) with the `simulateReadCounts` function.

```
> set.seed(1000)
> library(TCC)
> tcc <- simulateReadCounts(Ngene = 1000, PDEG = 0.2,
+                           DEG.assign = c(0.9, 0.1),
+                           DEG.foldchange = c(4, 4),
+                           replicates = c(3, 3))
> dim(tcc$count)

[1] 1000    6

> head(tcc$count)

       G1_rep1 G1_rep2 G1_rep3 G2_rep1 G2_rep2 G2_rep3
gene_1     168     104      62      38      24      35
gene_2      10       8      27       4       5       4
gene_3      64      94      81      15      57      12
gene_4     350     443     472     116     135     108
gene_5      92     263      60       7      15      22
gene_6     561     682     591      19      65     179

> tcc$group

        group
G1_rep1     1
G1_rep2     1
G1_rep3     1
G2_rep1     2
G2_rep2     2
G2_rep3     2
```

The simulation conditions for comparing two groups (G1 vs. G2) with biological replicates are as follows: (i) the number of genes is 1,000 (i.e., `Ngene = 1000`), (ii) the first 20% of genes are DEGs (`PDEG = 0.2`), (iii) the first 90% of the DEGs are up-regulated in G1, and the remaining 10% are up-regulated in G2 (`DEG.assign = c(0.9, 0.1)`), (iv) the levels of DE are four-fold in both groups (`DEG.foldchange = c(4, 4)`), and (v) there are a total of six samples (three biological replicates for G1 and three biological replicates for G2) (`replicates = c(3, 3)`). The variance of the NB distribution can be modeled as $V = \mu + \phi\mu^2$. The empirical distribution of the read counts for producing the mean (μ) and dispersion (ϕ) parameters of the model was obtained from *Arabidopsis* data (three biological replicates for each of the treated and non-treated groups) in `NBPSeq` (Di et al., 2011 [2]).

The `tcc$count` object is essentially the same as the `hypoData` object of `TCC`. The information about the simulation conditions can be viewed as follows.

```
> str(tcc$simulation)
```

```
List of 3
 $ trueDEG       : num [1:1000] 1 1 1 1 1 1 1 1 1 1 ...
 $ DEG.foldchange: num [1:1000, 1:6] 4 4 4 4 4 4 4 4 4 4 ...
 $ PDEG          : num [1:2] 0.18 0.02
```

Specifically, the entries for 0, 1, and 2 in the `tcc$simulation$trueDEG` object are for non-DEG, DEGs up-regulated in G1, and DEGs up-regulated in G2, respectively. The breakdowns for individual entries are the same as stated above: 800 entries are non-DEGs, 180 DEGs are up-regulated in G1, and 20 DEGs are up-regulated in G2.

```
> table(tcc$simulation$trueDEG)
```

```
  0   1   2
800 180  20
```

This information can be used to evaluate the performance of the DEGES-based normalization methods in terms of the sensitivity and specificity of the results of their DE analysis. A good normalization method coupled with a DE method such as the exact test (Robinson and Smyth, 2008 [9]) and the empirical Bayes (Hardcastle and Kelly, 2010) should produce well-ranked gene lists in which the true DEGs are top-ranked and non-DEGs are bottom-ranked when all genes are ranked according to the degree of DE. The ranked gene list after performing the DEGES/`edgeR`-`edgeR` combination can be obtained as follows.

```
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 1, FDR = 0.1, floorPDEG = 0.05)
> tcc <- estimateDE(tcc, test.method = "edger", FDR = 0.1)
> result <- getResult(tcc, sort = TRUE)
> head(result)
```

```
     gene_id   a.value   m.value      p.value      q.value rank estimatedDEG
74   gene_74  8.095342 -2.542249 1.632080e-10 1.286724e-07    1            1
185 gene_185  7.177367  2.407113 4.534722e-10 1.286724e-07    2            1
181 gene_181  7.861230  2.425154 5.023661e-10 1.286724e-07    3            1
11   gene_11  7.145969 -2.471502 5.786026e-10 1.286724e-07    4            1
113 gene_113 12.197408 -2.233912 6.433622e-10 1.286724e-07    5            1
138 gene_138  9.397467 -2.232690 1.118317e-09 1.863861e-07    6            1
```

We can now calculate the area under the `ROC` curve (i.e., AUC; $0 \leq \text{AUC} \leq 1$) between the ranked gene list and the truth (i.e., DEGs or non-DEGs) and thereby evaluate the sensitivity and specificity simultaneously. A well-ranked gene list should have a high AUC value (i.e., high sensitivity and specificity). The `calcAUCValue` function calculates the AUC value based on the information stored in the `TCC` class object.

```
> calcAUCValue(tcc)
```

```
[1] 0.8926656
```

This is essentially the same as

```
> AUC(rocdemo.sca(truth = as.numeric(tcc$simulation$trueDEG != 0),
+                 data = -tcc$stat$rank))
```

```
[1] 0.8926656
```

The following classic edgeR procedure (i.e., the TMM-edgeR combination) make it clear that the DEGES-based normalization method (i.e., the DEGES/edgeR pipeline) outperforms the default normalization method (i.e., TMM) implemented in edgeR.

```
> tcc <- calcNormFactors(tcc, norm.method = "tmm", iteration = 0)
> tcc <- estimateDE(tcc, test.method = "edger", FDR = 0.1)
> calcAUCValue(tcc)
```

```
[1] 0.8778063
```

The following is an alternative procedure for edgeR users.

```
> d <- DGEList(counts = tcc$count, group = tcc$group$group)
> d <- calcNormFactors(d)
> d$samples$norm.factors <- d$samples$norm.factors / mean(d$samples$norm.factors)
> d <- estimateCommonDisp(d)
> d <- estimateTagwiseDisp(d)
> result <- exactTest(d)
> result$table$PValue[is.na(result$table$PValue)] <- 1
> AUC(rocdemo.sca(truth = as.numeric(tcc$simulation$trueDEG != 0),
+                 data = -rank(result$table$PValue)))
```

```
[1] 0.8778063
```

As can be expected from the similarity of the normalization factors of DEGES/TbT (3.1.1) and DEGES/edgeR (3.1.2), the AUC value (0.8926656) of DEGES/edgeR is quite similar to the AUC value (0.8929750) of the original TbT method (i.e., DEGES/TbT):

```
> set.seed(1000)
> samplesize <- 100
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "bayseq",
+                        iteration = 1, samplesize = samplesize)
> tcc <- estimateDE(tcc, test.method = "edger", FDR = 0.1)
> calcAUCValue(tcc)
```

```
[1] 0.892975
```

5.2 Two-group data without replicates

Let us generate tag count data without replicates, such as those used in section 3.2 For simplicity, we first generate simulation data whose conditions are essentially the same as those in the previous section (i.e., 5.1), except for the number of replicates in each group: (i) the number of genes is 1,000 (i.e., `Ngene = 1000`), (ii) the first 20% of genes are DEGs (`PDEG = 0.2`), (iii) the first 90% of the DEGs are up-regulated in G1, and the remaining 10% are up-regulated in G2 (`DEG.assign = c(0.9, 0.1)`), (iv) the levels of DE are four-fold in both groups (`DEG.foldchange = c(4, 4)`), and (v) there are a total of two samples (one from G1 and the other from G2) (`replicates = c(1, 1)`).

```
> set.seed(1000)
> library(TCC)
> tcc <- simulateReadCounts(Ngene = 1000, PDEG = 0.2,
+                                DEG.assign = c(0.9, 0.1),
+                                DEG.foldchange = c(4, 4),
+                                replicates = c(1, 1))
> dim(tcc$count)
```

```
[1] 1000    2
```

```
> head(tcc$count)
```

```
       G1_rep1 G2_rep1
gene_1     168      29
gene_2      10       2
gene_3      64       2
gene_4     350      90
gene_5      92       5
gene_6     561      74
```

```
> tcc$group
```

```
        group
G1_rep1     1
G2_rep1     2
```

Now let us see how the DEGES/`DESeq-DESeq` combination with the original `DESeq-DESeq` combination performs. First, we calculate the AUC value for the ranked gene list obtained from the DEGES/`DESeq-DESeq` combination.

```
> tcc <- calcNormFactors(tcc, norm.method = "deseq", test.method = "deseq",
+                        iteration = 1, FDR = 0.1, floorPDEG = 0.05)
> tcc <- estimateDE(tcc, test.method = "deseq")
> calcAUCValue(tcc)
```

```
[1] 0.7845375
```

Next, we calculate the corresponding value using the original DESeq procedure (i.e., the DESeq-DESeq combination).

```
> tcc <- calcNormFactors(tcc, norm.method = "deseq", iteration = 0)
> tcc <- estimateDE(tcc, test.method = "deseq")
> calcAUCValue(tcc)
```

```
[1] 0.78265
```

It can be seen that the DEGES/DESeq-DESeq combination outperforms the original procedure under the given simulation conditions. The following is an alternative approach for DESeq users.

```
> cds <- newCountDataSet(tcc$count, tcc$group$group)
> cds <- estimateSizeFactors(cds)
> norm.factors <- sizeFactors(cds) / colSums(tcc$count)
> norm.factors <- norm.factors / mean(norm.factors)
> sizeFactors(cds) <- colSums(tcc$count) * norm.factors
> cds <- estimateDispersions(cds, method="blind", sharingMode="fit-only")
> result <- nbinomTest(cds, 1, 2)
> result$pval[is.na(result$pval)] <- 1
> AUC(rocdemo.sca(truth = as.numeric(tcc$simulation$trueDEG != 0),
+                 data = -rank(result$pval)))
```

```
[1] 0.78265
```

This procedure is completely the same as the one in TCC that gives normalization factors corresponding to those in edgeR for different packages. However, the following commands from the DESeq manual are of practical value because they give approximately the same AUC value as above.

```
> cds <- newCountDataSet(tcc$count, tcc$group$group)
> cds <- estimateSizeFactors(cds)
> cds <- estimateDispersions(cds, method="blind", sharingMode="fit-only")
> result <- nbinomTest(cds, 1, 2)
> result$pval[is.na(result$pval)] <- 1
> AUC(rocdemo.sca(truth = as.numeric(tcc$simulation$trueDEG != 0),
+                 data = -rank(result$pval)))
```

```
[1] 0.78265
```

5.3 Multi-group data with and without replicates

The simulateReadCounts function can generate simulation data with a more complex design. First, we generate a dataset consisting of three groups. The simulation conditions for this dataset are as follows: (i) the number of genes is 1,000 (i.e., Ngene = 1000), (ii) the first 30% of genes are DEGs (PDEG = 0.3), (iii) the breakdowns of the up-regulated DEGs are respectively 70%, 20%, and 10% in Groups 1-3 (DEG.assign = c(0.7, 0.2, 0.1)), (iv) the levels of DE are 3-, 10-, and 6-fold in individual groups (DEG.foldchange = c(3, 10, 6)), and (v) there are a total of nine libraries (2, 4, and 3 replicates for Groups 1-3) (replicates = c(2, 4, 3)).

```
> set.seed(1000)
> library(TCC)
> tcc <- simulateReadCounts(Ngene = 1000, PDEG = 0.3,
+                           DEG.assign = c(0.7, 0.2, 0.1),
+                           DEG.foldchange = c(3, 10, 6),
+                           replicates = c(2, 4, 3))
> dim(tcc$count)
```

```
[1] 1000    9
```

```
> tcc$group
```

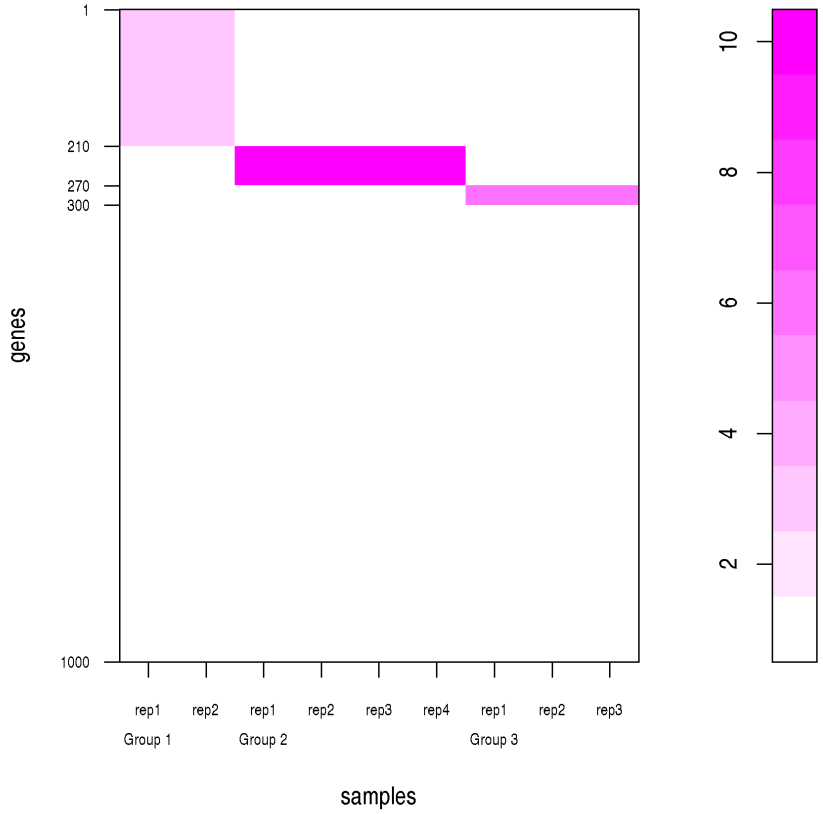
```
        group
G1_rep1     1
G1_rep2     1
G2_rep1     2
G2_rep2     2
G2_rep3     2
G2_rep4     2
G3_rep1     3
G3_rep2     3
G3_rep3     3
```

```
> head(tcc$count)
```

```
       G1_rep1 G1_rep2 G2_rep1 G2_rep2 G2_rep3 G2_rep4 G3_rep1 G3_rep2 G3_rep3
gene_1     126      86      17      38      24      35       4      19      71
gene_2       7       3       5       4       5       4       1       7       3
gene_3      48      17       3      15      57      12       4       9      10
gene_4     264     331     122     116     135     108      97      83      84
gene_5      69      51      32       7      15      22       4      33      11
gene_6     426     211      58      19      65     179      20     115      88
```

The pseudo-color image for the generated simulation data regarding the DEGs can be obtained from the `plotFCPseudocolor` function. The right bar (from white to magenta) indicates the degree of fold-change (FC). As expected, it can be seen that the first 210, 60, and 30 genes are up-regulated in G1, G2, and G3, respectively.

```
> plotFCPseudocolor(tcc)
```



Next, let us generate another dataset consisting of a total of eight groups. The simulation conditions for this dataset are as follows: (i) the number of genes is 10,000 (i.e., `Ngene = 10000`), (ii) the first 34% of genes are DEGs (`PDEG = 0.34`), (iii) the breakdowns of the up-regulated DEGs are respectively 10%, 30%, 5%, 10%, 5%, 21%, 9%, and 10% in Groups 1-8 (`DEG.assign = c(0.1, 0.3, 0.05, 0.1, 0.05, 0.21, 0.09, 0.1)`), (iv) the levels of DE are 3.1-, 13-, 2-, 1.5-, 9-, 5.6-, 4-, and 2-fold in individual groups (`DEG.foldchange = c(3.1, 13, 2, 1.5, 9, 5.6, 4, 2)`), and (v) there are a total of nine libraries (except for G3, none of the groups have replicates) (`replicates = c(1, 1, 2, 1, 1, 1, 1, 1)`).

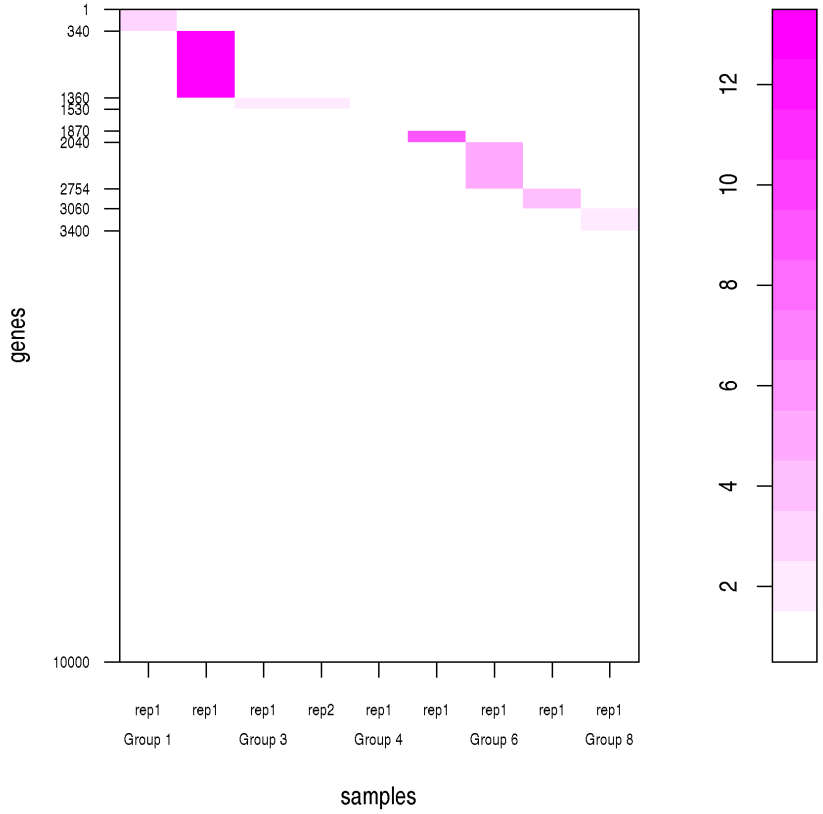
```
> set.seed(1000)
> library(TCC)
> tcc <- simulateReadCounts(Ngene = 10000, PDEG = 0.34,
+                DEG.assign = c(0.1, 0.3, 0.05, 0.1, 0.05, 0.21, 0.09, 0.1),
+                DEG.foldchange = c(3.1, 13, 2, 1.5, 9, 5.6, 4, 2),
+                replicates = c(1, 1, 2, 1, 1, 1, 1, 1))
> dim(tcc$count)

[1] 10000     9

> tcc$group
```

```
        group
G1_rep1     1
G2_rep1     2
G3_rep1     3
G3_rep2     3
G4_rep1     4
G5_rep1     5
G6_rep1     6
G7_rep1     7
G8_rep1     8
```

```
> head(tcc$count)
```

```
       G1_rep1 G2_rep1 G3_rep1 G3_rep2 G4_rep1 G5_rep1 G6_rep1 G7_rep1 G8_rep1
gene_1     253      32      16      14      27      93      25      17      26
gene_2      17       3       4       4       6       7       9       2       2
gene_3      51      17      10       9       9       4      15      10       9
gene_4     289      83     105      84     121      78      76     126      99
gene_5      43       4       4      11       0      19      34      42      15
gene_6     491      99     137     101      55     104     224     105      86
```

```
> plotFCPseudocolor(tcc)
```



This kind of simulation data may be useful for evaluating methods aimed at identifying tissue-specific (or tissue-selective) genes.

5.4 Other utilities

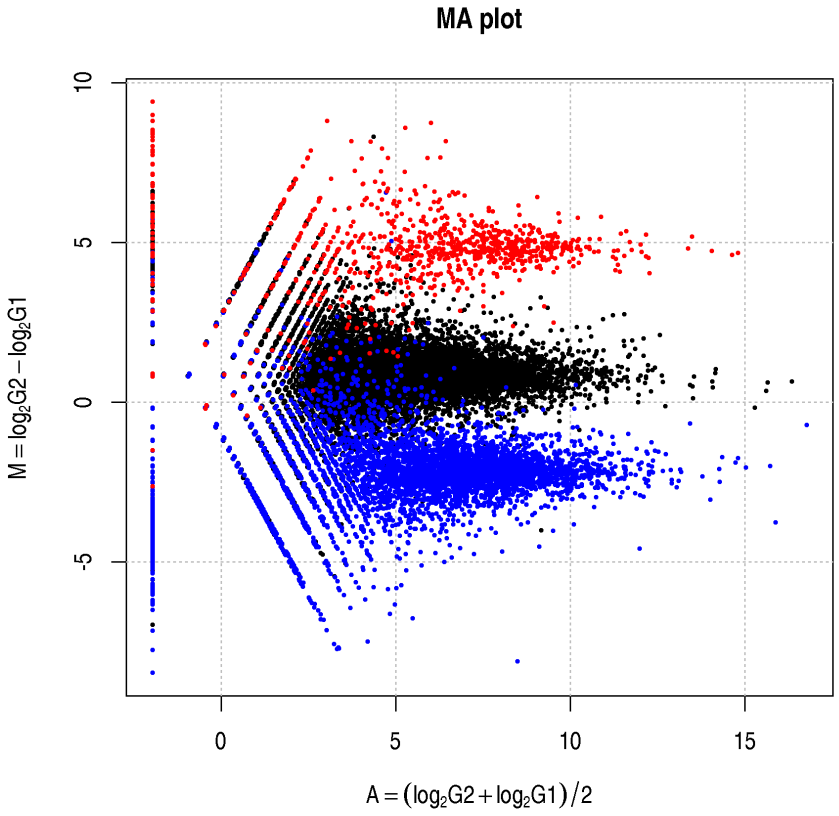
Recall that the simulation framework can handle different levels of DE for DEGs in individual groups, and the shape of the distribution for these DEGs is the same as that of non-DEGs. Let us confirm those distributions by introducing more drastic simulation conditions for comparing two groups (G1 vs. G2) with biological replicates; i.e., (i) the number of genes is 20,000 (i.e., `Ngene = 20000`), (ii) the first 30% of genes are DEGs (`PDEG = 0.30`), (iii) the first 85% of the DEGs are up-regulated in G1 and the remaining 15% are up-regulated in G2 (`DEG.assign = c(0.85, 0.15)`), (iv) the levels of DE are eight-fold in G1 and sixteen-fold in G2 (`DEG.foldchange = c(8, 16)`), and (v) there are a total of four samples (two biological replicates for G1 and two biological replicates for G2) (`replicates = c(2, 2)`).

```
> set.seed(1000)
> library(TCC)
> tcc <- simulateReadCounts(Ngene = 20000, PDEG = 0.30,
+                           DEG.assign = c(0.85, 0.15),
+                           DEG.foldchange = c(8, 16),
+                           replicates = c(2, 2))
> head(tcc$count)
```

```
        G1_rep1 G1_rep2 G2_rep1 G2_rep2
gene_1      368     243      67      39
gene_2       56      23       9       3
gene_3      140     147      11      18
gene_4      720     932     167      83
gene_5       56     165      50      11
gene_6      504     487      80     133
```

An M-A plot for the simulation data can be viewed as follows; the points for up-regulated DEGs in G1 and G2 are colored blue and red, respectively. The non-DEGs are in black:

```
> plot(tcc)
```



This plot is generated from simulation data that has been scaled in such a way that the library sizes of each sample are the same as the mean library size of the original data. That is,

```
> normalized.count <- getNormalizedData(tcc)
> colSums(normalized.count)
```

```
G1_rep1 G1_rep2 G2_rep1 G2_rep2
4474229 4474229 4474229 4474229
```

```
> colSums(tcc$count)
```

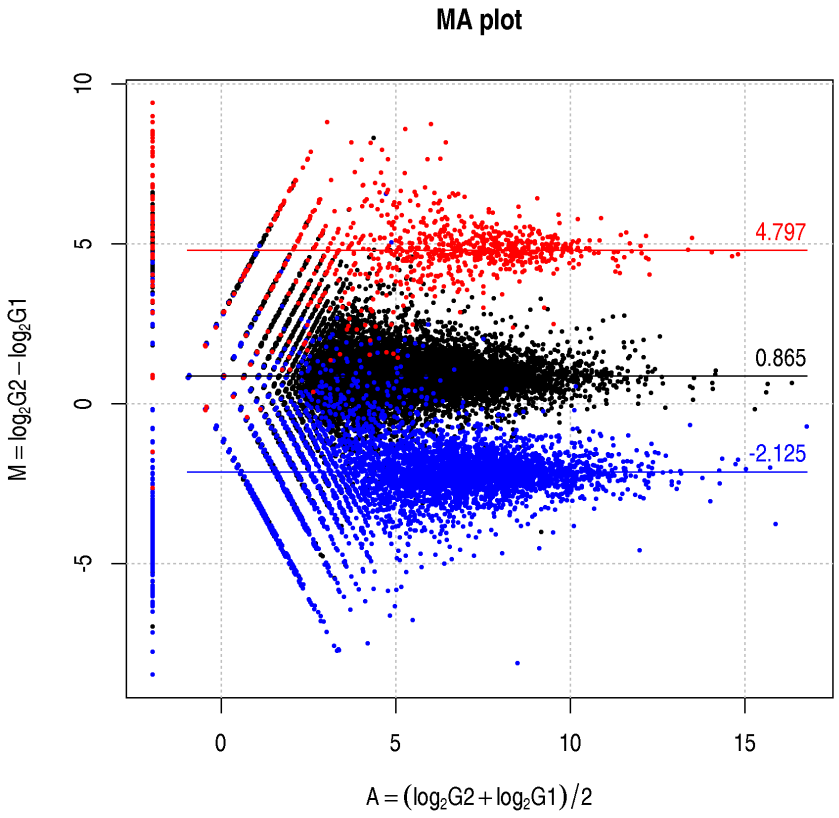
```
G1_rep1 G1_rep2 G2_rep1 G2_rep2
5704644 5813812 3108335 3270126
```

```
> mean(colSums(tcc$count))
```

```
[1] 4474229
```

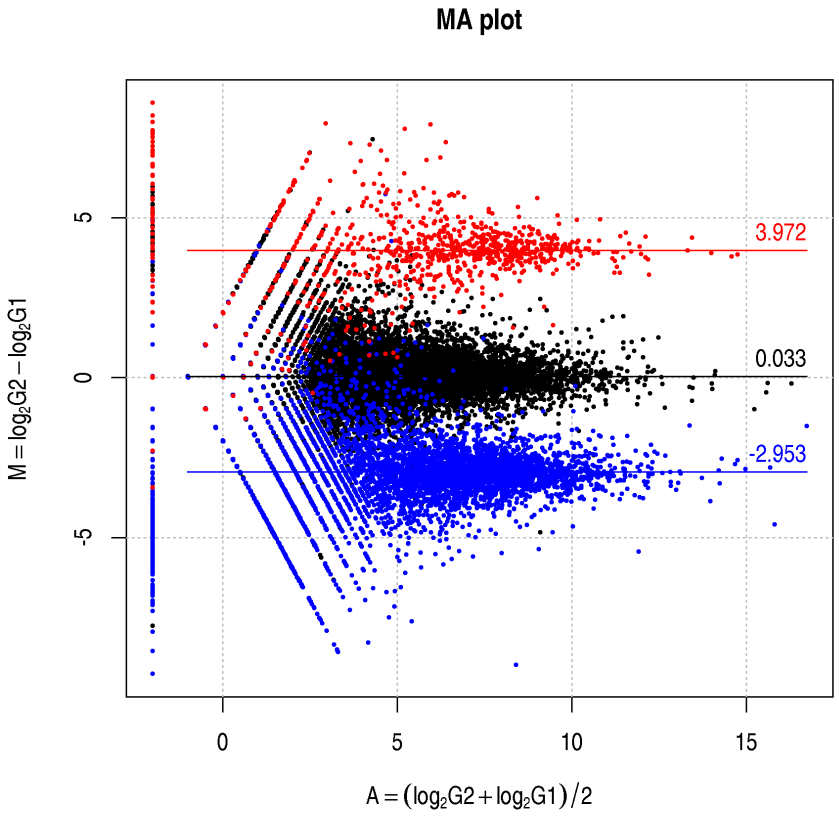
The summary statistics for non-DEGs and up-regulated DEGs in G1 and G2 are upshifted compared with the original intentions of the user (i.e., respective M values of 0, -3, and 4 for non-DEGs and up-regulated DEGs in G1 and G2). Indeed, the median values, indicated as horizontal lines, are respectively 0.865, -2.125, and 4.797 for non-DEGs and up-regulated DEGs in G1 and G2.

```
> plot(tcc, median.lines = TRUE)
```



These upshifted M values for non-DEGs can be modified after performing the iDEGES/`edgeR` normalization, e.g., the median M value ($= 0.033$) for non-DEGs based on the iDEGES/`edgeR`-normalized data is nearly zero.

```
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 3, FDR = 0.1, floorPDEG = 0.05)
> plot(tcc, median.line = TRUE)
```



The comparison of those values obtained from different normalization methods might be another evaluation metric.

6 Session info

```
> sessionInfo()


R version 3.0.1 RC (2013-05-12 r62739)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=C                 LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

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

other attached packages:
 [1] TCC_1.1.99         ROC_1.36.0         baySeq_1.14.0      edgeR_3.2.3
 [5] limma_3.16.3       DESeq_1.12.0       lattice_0.20-15    locfit_1.5-9.1
 [9] Biobase_2.20.0     BiocGenerics_0.6.0

loaded via a namespace (and not attached):
 [1] AnnotationDbi_1.22.5 DBI_0.2-7            IRanges_1.18.1
 [4] RColorBrewer_1.0-5   RSQLite_0.11.3       XML_3.96-1.1
 [7] annotate_1.38.0      genefilter_1.42.0    geneplotter_1.38.0
[10] grid_3.0.1           splines_3.0.1        stats4_3.0.1
[13] survival_2.37-4      tools_3.0.1          xtable_1.7-1
```

7 References

[1] Anders S and Huber W. Differential expression analysis for sequence count data. Genome Biol. 11(10): R106, 2010

[2] Di Y, Schafer DW, Cumbie JS, and Chang JH. The NBP negative binomial model for assessing differential gene expression from RNA-Seq. Stat Appl Genet Mol Biol. 10: art24, 2011

[3] Dillies MA, Rau A, Aubert J, Hennequet-Antier C, Jeanmougin M, Servant N, Keime C, Marot G, Castel D, Estelle J, Guernec G, Jagla B, Jouneau L, Laloë D, Le Gall C, Schaëffer B, Le Crom S, Guedj M, Jaffrézic F; on behalf of The French StatOmique Consortium. A comprehensive evaluation of normalization methods for Illumina high-throughput RNA sequencing data analysis. Brief Bioinform, in press

[4] Glaus P, Honkela A, and Rattray M. Identifying differentially expressed transcripts from RNA-seq data with biological variation. Bioinformatics 28(13): 1721-1728, 2012

[5] Hardcastle TJ and Kelly KA. `baySeq`: empirical Bayesian methods for identifying differential expression in sequence count data. BMC Bioinformatics 11: 422, 2010

[6] Kadota K, Nishiyama T, and Shimizu K. A normalization strategy for comparing tag count data. Algorithms Mol Biol. 7:5, 2012

[7] Robinson MD, McCarthy DJ, and Smyth GK. `edgeR`: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26(1): 139-140, 2010

[8] Robinson MD and Oshlack A. A scaling normalization method for differential expression analysis of RNA-seq data. Genome Biol., 11: R25, 2010

[9] Robinson MD and Smyth GK. Small-sample estimation of negative binomial dispersion, with applications to SAGE data. Biostatistics 9: 321-332, 2008

[10] Sun J, Nishiyama T, Shimizu K, and Kadota K. `TCC`: an R package for comparing tag count data with robust normalization strategies. submitted

[11] McCarthy DJ, Chen Y, Smyth GK. Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. Nucleic Acids Res. 40(10): 4288-4297, 2012