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: Jianqiang Sun (wukong@bi.a.u-tokyo.ac.jp),

Tomoaki Nishiyama (tomoakin@staff.kanazawa-u.ac.jp)

October 27, 2021

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 developed a multi-step normalization method (TbT; Kadota et al., 2012) 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 groups are extremely biased in their number. A major characteristic of **TCC** is to provide the DEGES-based normalization methods for several kinds of count data (two-group, multi-group, and so on) by virtue of the use of combinations of functions in other sophisticated packages (especially **edgeR**, 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 and **TCC** provides a simple unified interface to perform the robust normalization.

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., 2013) 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**, 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) recently reported that the normalization methods implemented in R packages (such as **edgeR** (Robinson et al., 2010), **DESeq** (Anders and Huber, 2010), and **baySeq** (Hardcastle and Kelly, 2010)) 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), in which the TMM normalization method (Robinson and Oshlack, 2010) is used in steps 1 and 3 and an empirical Bayesian method implemented in the **baySeq** package (Hardcastle and Kelly, 2010) 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 **DESeq2**. 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, 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**, **DESeq2**, 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:

```
> if (!requireNamespace("BiocManager", quietly=TRUE))
    > install.packages("BiocManager")
> BiocManager::install("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), which is a particular functionality of the **TCC** package (Sun et al., 2013), consists of the TMM normalization method (Robinson and Oshlack, 2010) implemented in the **edgeR** package (Robinson et al., 2010) and the empirical Bayesian method implemented in the **baySeq** package (Hardcastle and Kelly, 2010). Therefore, please cite the appropriate references when you publish your results.

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

1.3 Quick start

Let us begin by showing a example 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).

DE analysis for two-group count data with replicates by using the F-test (see `glmQLFTest` in **edgeR** package) 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., 2013), TMM (Robinson and Oshlack, 2010), and **edgeR** (Robinson et al., 2010). For details, see section 3.1.3 and 4.1.1.

```
> 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
1 gene_144 7.588253 -2.125607 8.062537e-06 0.002266315    1            1
2 gene_168 8.903810 -1.964742 1.131646e-05 0.002266315    2            1
3 gene_123 8.213453 -2.146315 1.441880e-05 0.002266315    3            1
4  gene_39 7.110128 -2.455020 1.507916e-05 0.002266315    4            1
5 gene_151 9.735347 -2.746644 1.597849e-05 0.002266315    5            1
6  gene_11 8.772814 -2.100665 1.664585e-05 0.002266315    6            1
```

2 Preparations

2.1 Preparing 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. There are many ways to obtain the count data from aligned read files such as BAM (Binary Alignment/Map). This includes the `islandCounts` function in **htSeqTools** (Planet et al., 2012), `summarizeOverlaps` in **GenomicRanges** (Lawrence et al., 2013), `qCount` in **QuasR**, and so on. For Windows end users, we recommend to use the **QuasR** package. It provides a comprehensive workflow for many kinds of NGS data including ChIP-seq, RNA-seq, smallRNA-seq, and BS-seq. The main functionalities are: (1) preprocessing raw sequence reads, (2) aligning reads to the reference genome or transcriptome using **Rbowtie**, and (3) producing count matrix from aligned reads. **TCC** uses the raw count data (a matrix of integer values) as input. As also clearly mentioned in **DESeq2**, the raw count data corresponds to a matrix of integer values. Please DO NOT use any normalized counts such as RPM (reads per million), CPM (counts per million), and RPKM.

2.2 Reading the count data

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 (G1_rep1, G1_rep2, and G1_rep3) and the remaining columns are from Group 2 (G2_rep1, G2_rep2, and 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)
```

2.3 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
```

2.4 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

This chapter describes the details of our robust normalization strategy (i.e., DEGES) implemented in **TCC**. Alternative DEGES procedures consisting of functions in other packages (**edgeR**, **DESeq2**, and **baySeq**) are also provided, enabling *advanced* users familiar with the existing packages can easily learn what is done in **TCC**. Note that *end* users, who just want to perform robust differential expression analysis using **TCC**, can skip this chapter (3 Normalization) and move from here to, for example, the next chapter (4 Differential expression). Note also 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). **TCC** can manipulate various kinds of experimental designs. Followings are some examples for individual designs. Appropriate sections should be referred for your specific experimental designs.

Table 1: 3.1 unpaired samples (two-group)

Label	Example 1	Example 2	Label	Example 3	Example 4
G1_rep1	G1(mouse_A)	Tumor(patient_A)	G1_rep1	G1(mouse_A)	Tumor(patient_A)
G1_rep2	G1(mouse_B)	Tumor(patient_B)	G1_rep2	G1(mouse_B)	Tumor(patient_B)
G1_rep3	G1(mouse_C)	Tumor(patient_C)	G2_rep1	G2(mouse_D)	Normal(patient_G)
G2_rep1	G2(mouse_D)	Normal(patient_G)	G2_rep2	G2(mouse_E)	Normal(patient_H)
G2_rep2	G2(mouse_E)	Normal(patient_H)			
G2_rep3	G2(mouse_F)	Normal(patient_K)			

Table 2: 3.2 paired samples (two-group)

Label	Example 1	Example 2	Label	Example 3	Example 4
G1_rep1	G1(mouse_A)	Tumor(patient_G)	G1_rep1	G1(mouse_B)	Tumor(patient_J)
G1_rep2	G1(mouse_B)	Tumor(patient_H)	G1_rep2	G1(mouse_C)	Tumor(patient_K)
G1_rep3	G1(mouse_C)	Tumor(patient_K)	G2_rep1	G2(mouse_B)	Normal(patient_J)
G2_rep1	G2(mouse_A)	Normal(patient_G)	G2_rep2	G2(mouse_C)	Normal(patient_K)
G2_rep2	G2(mouse_B)	Normal(patient_H)			
G2_rep3	G2(mouse_C)	Normal(patient_K)			

Table 3: 3.3 unpaired samples (multi-group)

Label	Example 1	Example 2	Label	Example 3	Example 4
G1_rep1	G1(mouse_A)	Kidney(sample_A)	G1_rep1	G1(mouse_A)	Liver(sample_G)
G1_rep2	G1(mouse_B)	Kidney(sample_B)	G1_rep2	G1(mouse_B)	Liver(sample_H)
G1_rep3	G1(mouse_C)	Kidney(sample_C)	G2_rep1	G2(mouse_D)	Brain(sample_Y)
G2_rep1	G2(mouse_D)	Liver(sample_G)	G2_rep2	G2(mouse_E)	Brain(sample_Z)
G2_rep2	G2(mouse_E)	Liver(sample_H)	G3_rep1	G3(mouse_U)	Kidney(sample_B)
G2_rep3	G2(mouse_F)	Liver(sample_K)	G3_rep2	G3(mouse_T)	Kidney(sample_C)
G3_rep1	G3(mouse_G)	Brain(sample_X)			
G3_rep2	G3(mouse_H)	Brain(sample_Y)			
G3_rep3	G3(mouse_I)	Brain(sample_Z)			

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). 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/DESeq2).

3.1.1 DEGES/TbT

The DEGES/TbT (Kadota et al., 2012) with default parameter settings can be performed 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 = "bayseq",
+                        iteration = 1, samplesize = samplesize)
```

Note that a smaller sampling size (i.e., `samplesize = 100`) 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). 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
0.8811041 0.8456380 0.8402077 1.0887419 1.1534638 1.1908445
```

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

```
   user  system elapsed
   3.05    0.05    3.09
```

Of course, the procedure can be performed by using functions in **edgeR** and **baySeq**, instead of using the `calcNormFactors` function in **TCC**. The `calcNormFactors` 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 <- 100
> group <- c(1, 1, 1, 2, 2, 2)
> floorPDEG <- 0.05
> FDR <- 0.1
> ### 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@annotation <- data.frame(rowID = 1:nrow(hypoData))
> cD <- getPriors.NB(cD, samplesize = samplesize, estimation = "QL", cl = NULL)
> cD <- getLikelihoods(cD, pET = "BIC", cl = NULL)
```

```
.
```

```
> result <- topCounts(cD, group = "DE", number = nrow(hypoData))
> result <- result[order(result$rowID), ]
> pval <- result$FWER.DE
> qval <- result$FDR.DE
> if (sum(qval < FDR) > (floorPDEG * nrow(hypoData))) {
+   is.DEG <- as.logical(qval < FDR)
+ } else {
+   is.DEG <- as.logical(rank(pval, ties.method = "min") <=
+                        nrow(hypoData) * floorPDEG)
+ }
> ### STEP 3 ###
> d <- DGEList(count = hypoData[!is.DEG, ], group = group)
> d <- calcNormFactors(d)
> norm.factors <- d$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
0.8811041 0.8456380 0.8402077 1.0887419 1.1534638 1.1908445
```

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
0.8807362 0.8603783 0.8464462 1.0841895 1.1447986 1.1834512


> tcc$DEGES$execution.time


   user  system elapsed
   0.24    0.00    0.23
```

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 `calcNormFactors` 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 ###
> design <- model.matrix(~group)
> d <- estimateDisp(d, design)
> fit <- glmQLFit(d, design)
> out <- glmQLFTest(fit, coef = 2)
> pval <- out$table$PValue
> qval <- p.adjust(pval, method = "BH")
> if (sum(qval < FDR) > (floorPDEG * nrow(hypoData))) {
+   is.DEG <- as.logical(qval < FDR)
+ } else {
+   is.DEG <- as.logical(rank(pval, 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
0.8807362 0.8603783 0.8464462 1.0841895 1.1447986 1.1834512
```

3.1.3 iDEGES/edgeR

Our multi-step normalization can be repeated until the calculated normalization factors converge (Kadota et al., 2012). 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
0.8704550 0.8444514 0.8456197 1.0777233 1.1495631 1.2121876
```

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

```
   user  system elapsed
   0.66    0.00    0.65
```

3.1.4 DEGES/DESeq2

The DEGES pipeline can also be performed by using only the functions in the **DESeq2** package. Similar to the **DESeq** case above, this DESeq2-DESeq2-DESeq2 pipeline (DEGES/DESeq2) 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 = "deseq2", test.method = "deseq2",
+                        iteration = 1, FDR = 0.1, floorPDEG = 0.05)
> tcc$norm.factors
```

```
  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3
0.8804811 0.8712588 0.8207842 1.0784376 1.1570976 1.1919407
```

```
> tcc$DEGES$execution.time


   user  system elapsed
   3.85    0.04    3.89
```

For **DESeq2** users, we also provide commands, consisting of functions in **DESeq2**, to perform the DEGES/DESeq2 pipeline without **TCC**. The `calcNormFactors` function together with the above arguments can be regarded as a wrapper function for the following commands.

```
> library(TCC)
> data(hypoData)
> FDR <- 0.1
> floorPDEG <- 0.05
> group <- factor(c(1, 1, 1, 2, 2, 2))
> cl <- data.frame(group = group)
> design <- formula(~ group)
> dds <- DESeqDataSetFromMatrix(countData = hypoData, colData = cl,
+                               design = design)
> ### STEP 1 ###
> dds <- estimateSizeFactors(dds)
> sizeFactors(dds) <- sizeFactors(dds) / mean(sizeFactors(dds))
> ### STEP 2 ###
> dds <- estimateDispersions(dds)
> dds <- nbinomWaldTest(dds)
> result <- results(dds)
> result$pvalue[is.na(result$pvalue)] <- 1
> pval <- result$pvalue
> qval <- p.adjust(pval, method = "BH")
> if (sum(qval < FDR) > (floorPDEG * nrow(hypoData))) {
+   is.DEG <- as.logical(qval < FDR)
+ } else {
+   is.DEG <- as.logical(rank(pval, ties.method = "min") <=
+                        nrow(hypoData) * floorPDEG)
+ }
> ### STEP 3 ###
> dds <- DESeqDataSetFromMatrix(countData = hypoData[!is.DEG, ], colData = cl,
+                               design = design)
> dds <- estimateSizeFactors(dds)
> norm.factors <- sizeFactors(dds) / colSums(hypoData)
> norm.factors <- norm.factors / mean(norm.factors)
> norm.factors


  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3
0.8804811 0.8712588 0.8207842 1.0784376 1.1570976 1.1919407
```

3.2 Normalization of two-group count data without replicates (paired)

edgeR and **DESeq2** employs generalized linear models (GLMs) which can apply to detect DEGs from paired two-group count data. When obtaining normalization factors from paired two group samples, users can select a total of four combinations (two normalization methods $\times$ two DEG identification methods) coupled with an arbitrary number of iterations ($n = 0, 1, 2, \cdots, 100$) in our DEGES-based normalization pipeline. The analysis for paired samples can be performed by indicating (1) the pair information when constructing the `TCC` class object and (2) `paired = TRUE` when performing the `calcNormFactors` function. For analyzing paired data, we here use the hypothetical count data (`hypoData`; for details see 2.2) by changing the label information, i.e.,

```
> library(TCC)
> data(hypoData)
> colnames(hypoData) <- c("T_dogA", "T_dogB", "T_dogC",
+                         "N_dogA", "N_dogB", "N_dogC")
> head(hypoData)
```

```
       T_dogA T_dogB T_dogC N_dogA N_dogB N_dogC
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
```

This count data consists of three individuals (or sibships), dogA, dogB, and dogC. "T" and "N" indicate tumor and normal tissues, respectively.

3.2.1 DEGES/edgeR

The DEGES/edgeR pipeline for two-group paired data can be performed as follows:

```
> library(TCC)
> data(hypoData)
> colnames(hypoData) <- c("T_dogA", "T_dogB", "T_dogC",
+                         "N_dogA", "N_dogB", "N_dogC")
> group <- c(1, 1, 1, 2, 2, 2)
> pair <- c(1, 2, 3, 1, 2, 3)
> cl <- data.frame(group = group, pair = pair)
> tcc <- new("TCC", hypoData, cl)
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 1, FDR = 0.1, floorPDEG = 0.05, paired = TRUE)
> tcc$norm.factors
```

```
   T_dogA    T_dogB    T_dogC    N_dogA    N_dogB    N_dogC
0.8814045 0.8628974 0.8453069 1.0821227 1.1464102 1.1818582
```

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

```
> library(TCC)
> data(hypoData)
> colnames(hypoData) <- c("T_dogA", "T_dogB", "T_dogC",
+                         "N_dogA", "N_dogB", "N_dogC")
> group <- factor(c(1, 1, 1, 2, 2, 2))
> pair <- factor(c(1, 2, 3, 1, 2, 3))
> design <- model.matrix(~ group + pair)
> coef <- 2
> FDR <- 0.1
> floorPDEG <- 0.05
> d <- DGEList(counts = hypoData, group = group)
> ### STEP 1 ###
> d <- calcNormFactors(d)
> ### STEP 2 ###
> d <- estimateDisp(d, design)
> fit <- glmQLFit(d, design)
> out <- glmQLFTest(fit, coef = coef)
> pval <- out$table$PValue
> qval <- p.adjust(pval, method = "BH")
```

```
> if (sum(qval < FDR) > (floorPDEG * nrow(hypoData))){
+   is.DEG <- as.logical(qval < FDR)
+ } else {
+   is.DEG <- as.logical(rank(pval, 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

   T_dogA    T_dogB    T_dogC    N_dogA    N_dogB    N_dogC
0.8814045 0.8628974 0.8453069 1.0821227 1.1464102 1.1818582
```

3.3 Normalization of multi-group count data with replicates

Many R packages (including **edgeR**, 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, and 3.3.2 DEGES/edgeR).

3.3.1 DEGES/TbT

The DEGES/TbT pipeline for multi-group data is essentially the same as those for two-group data with or 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).

```
> 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
1.0310998 0.9182698 0.7988597 0.8266250 1.2152515 1.1880931 0.9788746 1.0061709
  G3_rep3
1.0367556
```

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 DEGES/edgeR pipeline with the two models (`design` and `coef`) can be performed 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
1.0259958 0.9108415 0.7967735 0.8300224 1.1798804 1.1757521 0.9986086 1.0282825
  G3_rep3
1.0538432
```

The two models (`design` and `coef`) will automatically be generated when performing the following `calcNormFactors` function if those models are not explicitly indicated. That is

```
> library(TCC)
> data(hypoData_mg)
> 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 = "edger",
+                        iteration = 1)
> tcc$norm.factors
```

```
  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3   G3_rep1   G3_rep2
1.0259958 0.9108415 0.7967735 0.8300224 1.1798804 1.1757521 0.9986086 1.0282825
  G3_rep3
1.0538432
```

For **edgeR** users, we provide commands, consisting of functions in **edgeR**, to perform the DEGES/edgeR pipeline without **TCC**. The `calcNormFactors` 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:ncol(design)
> d <- DGEList(counts = hypoData_mg, group = group)
> ### STEP 1 ###
> d <- calcNormFactors(d)
> ### STEP 2 ###
> d <- estimateDisp(d, design)
> fit <- glmQLFit(d, design)
> out <- glmQLFTest(fit, coef = coef)
> result <- as.data.frame(topTags(out, n = nrow(hypoData_mg)))
> result <- result$table[rownames(hypoData_mg), ]
> pval <- out$table$PValue
> qval <- p.adjust(pval, method = "BH")
> if (sum(qval < FDR) > (floorPDEG * nrow(hypoData_mg))) {
+   is.DEG <- as.logical(qval < FDR)
+ } else  {
+   is.DEG <- as.logical(rank(pval, ties.method = "min") <=
+                        nrow(hypoData_mg) * floorPDEG)
+ }
> ### STEP 3 ###
> d <- DGEList(counts = hypoData_mg[!is.DEG, ], group = group)
> d <- calcNormFactors(d)
> norm.factors <- d$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
1.0259958 0.9108415 0.7967735 0.8300224 1.1798804 1.1757521 0.9986086 1.0282825
  G3_rep3
1.0538432
```

3.4 Retrieving normalized data

Similar functions for calculating normalization factors are the `calcNormFactors` function in **edgeR** and the `estimateSizeFactors` function in **DESeq2**. Note that the terminology used in **DESeq2** (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 both **DESeq2** 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), 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); 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. The DEGES/edgeR-normalized data can be retrieved as follows.

```
> 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
20.15002 19.69267 19.40289 21.26696 21.68738 20.39778


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


[1] 19.40289 21.68738
```

For comparison, the summary statistics for TMM-normalized data produced using the original normalization method (i.e., TMM) in **edgeR** can be 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
> 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


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


[1] 18.59060 22.98591
```

It is obvious that the summary statistics (ranging from 19.40289 to 21.68738) from DEGES/edgeR-normalized data are closer to the truth (i.e., ranging from 20.0 to 21.0) than those (ranging from 18.59060 to 22.98591 from TMM-normalized data.

3.4.2 Retrieving two-group DEGES/edgeR-normalized data without replicates (paired)

We here analyze the `hypoData` object provided in **TCC**. As described in 3.2, we regard `hypoData` as a hypothetical paired data. The DEGES/edgeR-normalized data can be retrieved as follows.

```
> library(TCC)
> data(hypoData)
> colnames(hypoData) <- c("T_dogA", "T_dogB", "T_dogC",
+                         "N_dogA", "N_dogB", "N_dogC")
> nonDEG <- 201:1000
> group <- c(1, 1, 1, 2, 2, 2)
> pair <- c(1, 2, 3, 1, 2, 3)
> cl <- data.frame(group = group, pair = pair)
> tcc <- new("TCC", hypoData, cl)
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 1, FDR = 0.1, floorPDEG = 0.05, paired = TRUE)
> normalized.count <- getNormalizedData(tcc)
> head(normalized.count, n = 4)

          T_dogA    T_dogB    T_dogC    N_dogA    N_dogB    N_dogC
gene_1   33.3971  44.18299 118.52741  16.23575  14.43918  29.61920
gene_2  351.6518 380.95556  21.37380  36.53044  25.78425  69.45192
gene_3 1123.7142 902.31483 961.82082 379.51064 495.05760 244.10309
gene_4    0.0000   0.00000  42.74759  18.26522   0.00000   0.00000

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

  T_dogA   T_dogB   T_dogC   N_dogA   N_dogB   N_dogC
20.13649 19.63688 19.43072 21.30942 21.65877 20.42704

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

[1] 19.43072 21.65877
```

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

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

  T_dogA   T_dogB   T_dogC   N_dogA   N_dogB   N_dogC
19.35893 19.01078 18.59060 22.98591 22.16273 21.00685

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

[1] 18.59060 22.98591
```

It is obvious that the summary statistics (ranging from 19.43072 to 21.65877) from DEGES/edgeR-normalized data are closer to the truth (i.e., ranging from 20.0 to 21.0) than those (ranging from 18.59060 to 22.98591) from TMM-normalized data.

3.4.3 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)
> 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
13.88031 13.09456 13.95500 13.06733 13.28704 13.85287 14.28137 14.82470
 G3_rep3
15.25078
```

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

```
[1] 13.06733 15.25078
```

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)
> d <- DGEList(tcc$count, group = group)
> d <- calcNormFactors(d)
> norm.factors <- d$samples$norm.factors
> effective.libsizes <- colSums(hypoData) * norm.factors
> normalized.count <- sweep(hypoData, 2,
+                     mean(effective.libsizes) / effective.libsizes, "*")
> apply(normalized.count[nonDEG, ], 2, median)
```

```
  T_dogA   T_dogB   T_dogC   N_dogA   N_dogB   N_dogC
21.12359 19.78313 19.69837 20.59855 20.66165 19.89551
```

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

```
[1] 19.69837 21.12359
```

It is obvious that the summary statistics (ranging from 13.06733 to 15.25078) from iDEGES/edgeR-normalized data are closer to the truth (i.e., ranging from 13.0 to 15.0) than those (ranging from 19.69837 to 21.12359) 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. Similar to the usage in the `calcNormFactors` function with the `test.method` argument in **TCC**, those DE methods in **edgeR**, and **baySeq** can be performed by using the `estimateDE` function with `test.method = "edger"`, `"deseq2"`, and `"bayseq"`, respectively. Here, we show some examples of DE analysis for two-group data with replicates (4.1), two-group data without replicates (**??**), paired 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
1 gene_144 7.588253 -2.125607 8.062537e-06 0.002266315    1            1
2 gene_168 8.903810 -1.964742 1.131646e-05 0.002266315    2            1
3 gene_123 8.213453 -2.146315 1.441880e-05 0.002266315    3            1
4  gene_39 7.110128 -2.455020 1.507916e-05 0.002266315    4            1
5 gene_151 9.735347 -2.746644 1.597849e-05 0.002266315    5            1
6  gene_11 8.772814 -2.100665 1.664585e-05 0.002266315    6            1
```

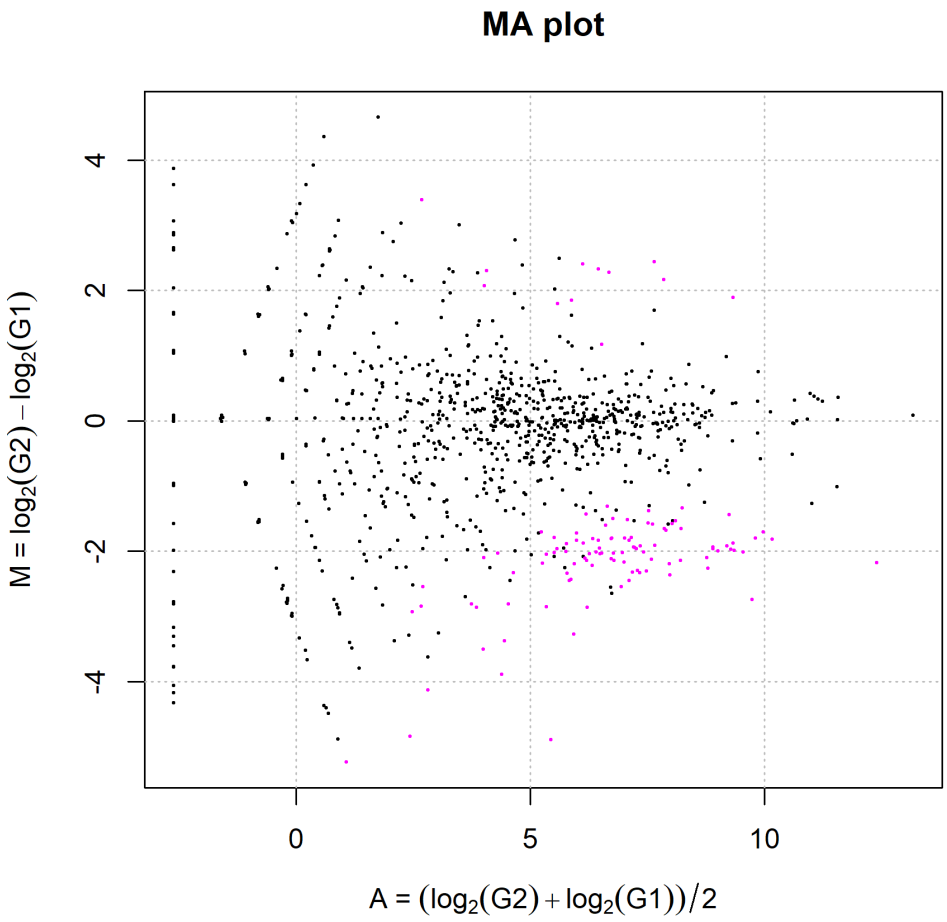
The DE results can be broken down as follows.

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

```
  0   1
888 112
```

This means 888 non-DEGs and 112 DEGs satisfy FDR < 0.1. The `plot` function generates an M-A plot, where "M" indicates the log-ratio (i.e., $\mathrm{M} = \log_2 \mathrm{G2} - \log_2 \mathrm{G1}$) and "A" indicates average read count (i.e., $\mathrm{A} = (\log_2 \mathrm{G2} + \log_2 \mathrm{G1})/2$), from the normalized count data. The magenta points indicate the identified DEGs at FDR < 0.1.

```
> plot(tcc)
```



Since we know the true information for `hypoData` (i.e., 200 DEGs and 800 non-DEGs), we can calculate the area under the ROC curve (i.e., AUC; $0 \leq \text{AUC} \leq 1$) between the ranked gene list and the truth 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).

```
> library(ROC)
> truth <- c(rep(1, 200), rep(0, 800))
> AUC(rocdemo.sca(truth = truth, data = -tcc$stat$rank))
```

```
[1] 0.8834219
```

For comparison, the DE results from the original procedure in **edgeR** can be obtained as follows.

```
> library(TCC)
> data(hypoData)
> group <- c(1, 1, 1, 2, 2, 2)
> design <- model.matrix(~ as.factor(group))
> d <- DGEList(count = hypoData, group = group)
> d <- calcNormFactors(d)
> d <- estimateDisp(d, design)
> fit <- glmQLFit(d, design)
> out <- glmQLFTest(fit, coef = 2)
> result <- as.data.frame(topTags(out, n = nrow(hypoData), sort.by = "PValue"))
> head(result)
```

```
             logFC    logCPM        F       PValue         FDR
gene_144 -1.979269 10.925554 96.40845 1.375224e-05 0.003246231
gene_189  2.545307  9.661868 94.65954 1.467659e-05 0.003246231
gene_168 -1.820367 12.186563 90.96828 1.690004e-05 0.003246231
gene_11  -1.956542 12.094762 82.87398 2.348301e-05 0.003246231
gene_39  -2.305902 10.555814 80.33504 2.619616e-05 0.003246231
gene_151 -2.599989 13.268812 79.94060 2.665255e-05 0.003246231
```

This is the same as

```
> library(TCC)
> data(hypoData)
> group <- c(1, 1, 1, 2, 2, 2)
> tcc <- new("TCC", hypoData, group)
> tcc <- calcNormFactors(tcc, norm.method = "tmm", iteration = 0)
> 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
1 gene_144 7.591264 -1.980620 1.375224e-05 0.003246231    1            1
2 gene_189 6.127498  2.550316 1.467659e-05 0.003246231    2            1
3 gene_168 8.905896 -1.820884 1.690004e-05 0.003246231    3            1
4  gene_11 8.774575 -1.957134 2.348301e-05 0.003246231    4            1
5  gene_39 7.114394 -2.308040 2.619616e-05 0.003246231    5            1
6 gene_151 9.740529 -2.600366 2.665255e-05 0.003246231    6            1
```

```
> AUC(rocdemo.sca(truth = truth, data = -tcc$stat$rank))
```

```
[1] 0.8649281
```

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
1 gene_144 7.591264 -1.980620 1.375224e-05 0.003246231    1            1
2 gene_189 6.127498  2.550316 1.467659e-05 0.003246231    2            1
3 gene_168 8.905896 -1.820884 1.690004e-05 0.003246231    3            1
4  gene_11 8.774575 -1.957134 2.348301e-05 0.003246231    4            1
5  gene_39 7.114394 -2.308040 2.619616e-05 0.003246231    5            1
6 gene_151 9.740529 -2.600366 2.665255e-05 0.003246231    6            1
```

The DE results can be broken down as follows.

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

```
  0   1
888 112
```

This means 888 non-DEGs and 112 DEGs satisfy FDR < 0.1. The `plot` function generates an M-A plot, where "M" indicates the log-ratio (i.e., $\mathrm{M} = \log_2 \mathrm{G2} - \log_2 \mathrm{G1}$) and "A" indicates average read count (i.e., $\mathrm{A} = (\log_2 \mathrm{G2} + \log_2 \mathrm{G1})/2$), from the normalized count data. The magenta points indicate the identified DEGs at FDR < 0.1.

```
> plot(tcc)
```

Since we know the true information for `hypoData` (i.e., 200 DEGs and 800 non-DEGs), we can calculate the area under the ROC curve (i.e., AUC; $0 \leq \text{AUC} \leq 1$) between the ranked gene list and the truth 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).

```
> library(ROC)
> truth <- c(rep(1, 200), rep(0, 800))
> AUC(rocdemo.sca(truth = truth, data = -tcc$stat$rank))
```

```
[1] 0.8649281
```

4.1.2 DESeq2 coupled with iDEGES/DESeq2 normalization

For comparison, the DE results from the original procedure in **DESeq2** can be obtained as follows.

```
> library(TCC)
> data(hypoData)
> group <- factor(c(1, 1, 1, 2, 2, 2))
> cl <- data.frame(group = group)
> design <- formula(~ group)
```

```
> dds <- DESeqDataSetFromMatrix(countData = hypoData, colData = cl,
+                               design = design)
> dds <- estimateSizeFactors(dds)
> sizeFactors(dds) <- sizeFactors(dds) / mean(sizeFactors(dds))
> dds <- estimateDispersions(dds)
> dds <- nbinomWaldTest(dds)
> result <- results(dds)
> head(result[order(result$pvalue),])
```

```
log2 fold change (MLE): group 2 vs 1
Wald test p-value: group 2 vs 1
DataFrame with 6 rows and 6 columns
          baseMean log2FoldChange     lfcSE      stat      pvalue        padj
         <numeric>      <numeric> <numeric> <numeric>   <numeric>   <numeric>
gene_168   577.362       -1.81130  0.188923  -9.58750 9.02463e-22 7.58971e-19
gene_144   239.748       -1.97304  0.220284  -8.95678 3.34295e-19 1.40571e-16
gene_11    541.281       -1.94638  0.230809  -8.43287 3.37280e-17 9.45507e-15
gene_143   752.478       -1.81641  0.218271  -8.32181 8.66293e-17 1.82138e-14
gene_115   574.796       -1.78980  0.216748  -8.25749 1.48766e-16 2.50224e-14
gene_109   763.275       -1.72650  0.214623  -8.04432 8.67290e-16 1.21565e-13
```

```
> library(ROC)
> truth <- c(rep(1, 200), rep(0, 800))
> result$pvalue[is.na(result$pvalue)] <- 1
> AUC(rocdemo.sca(truth = truth, data = -rank(result$pvalue)))
```

```
[1] 0.8632313
```

This is the same as

```
> library(TCC)
> data(hypoData)
> group <- c(1, 1, 1, 2, 2, 2)
> tcc <- new("TCC", hypoData, group)
> tcc <- calcNormFactors(tcc, norm.method = "deseq2", iteration = 0)
> tcc <- estimateDE(tcc, test.method = "deseq2", FDR = 0.1)
> result <- getResult(tcc, sort = TRUE)
> head(result)
```

```
   gene_id  a.value   m.value      p.value      q.value rank estimatedDEG
1 gene_168 8.905999 -1.811262 9.024627e-22 9.024627e-19    1            1
2 gene_144 7.591478 -1.973100 3.342948e-19 1.671474e-16    2            1
3  gene_11 8.774276 -1.946228 3.372796e-17 1.124265e-14    3            1
4 gene_143 9.286710 -1.816512 8.662935e-17 2.165734e-14    4            1
5 gene_115 8.905535 -1.789730 1.487657e-16 2.975314e-14    5            1
6 gene_109 9.331890 -1.726446 8.672896e-16 1.445483e-13    6            1
```

```
> library(ROC)
> truth <- c(rep(1, 200), rep(0, 800))
> AUC(rocdemo.sca(truth = truth, data = -tcc$stat$rank))
```

```
[1] 0.8632219
```

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

```
  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3
0.9271692 0.9147550 0.8753592 1.0217885 1.1069052 1.1540229
```

As described in 3.4, the size factors termed in **DESeq2** is comparable to the *normalized* effective library sizes termed in **TCC** and **edgeR**. The effective library size in **edgeR** is calculated as the library size multiplied by the normalization factor. The normalization factors and effective library sizes in **DESeq2** can be retrieved as follows.

```
> library(TCC)
> data(hypoData)
> group <- factor(c(1, 1, 1, 2, 2, 2))
> cl <- data.frame(group = group)
> design <- formula(~ group)
> dds <- DESeqDataSetFromMatrix(countData = hypoData, colData = cl,
+                               design = design)
> dds <- estimateSizeFactors(dds)
> size.factors <- sizeFactors(dds)
> size.factors
```

```
  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3
1.0830730 1.0919605 1.0779951 0.9411006 0.9468033 0.9668911
```

```
> hoge <- size.factors / colSums(hypoData)
> norm.factors <- hoge / mean(hoge)
> norm.factors
```

```
  G1_rep1   G1_rep2   G1_rep3   G2_rep1   G2_rep2   G2_rep3
0.9271692 0.9147550 0.8753592 1.0217885 1.1069052 1.1540229
```

```
> ef.libsizes <- norm.factors * colSums(hypoData)
> ef.libsizes
```

```
 G1_rep1  G1_rep2  G1_rep3  G2_rep1  G2_rep2  G2_rep3
131822.1 132903.8 131204.1 114542.5 115236.6 117681.5
```

Note that the following commands should be the simplest procedure provided in **DESeq2**.

```
> library(TCC)
> data(hypoData)
> group <- factor(c(1, 1, 1, 2, 2, 2))
> cl <- data.frame(group = group)
> design <- formula(~ group)
> dds <- DESeqDataSetFromMatrix(countData = hypoData, colData = cl,
+                               design = design)
> dds <- estimateSizeFactors(dds)
> sizeFactors(dds) <- sizeFactors(dds) / mean(sizeFactors(dds))
> dds <- estimateDispersions(dds)
> dds <- nbinomWaldTest(dds)
> result <- results(dds)
> head(result[order(result$pvalue),])
```

```
log2 fold change (MLE): group 2 vs 1
Wald test p-value: group 2 vs 1
DataFrame with 6 rows and 6 columns
          baseMean log2FoldChange     lfcSE      stat      pvalue        padj
         <numeric>      <numeric> <numeric> <numeric>   <numeric>   <numeric>
gene_168   577.362       -1.81130  0.188923  -9.58750 9.02463e-22 7.58971e-19
gene_144   239.748       -1.97304  0.220284  -8.95678 3.34295e-19 1.40571e-16
gene_11    541.281       -1.94638  0.230809  -8.43287 3.37280e-17 9.45507e-15
gene_143   752.478       -1.81641  0.218271  -8.32181 8.66293e-17 1.82138e-14
gene_115   574.796       -1.78980  0.216748  -8.25749 1.48766e-16 2.50224e-14
gene_109   763.275       -1.72650  0.214623  -8.04432 8.67290e-16 1.21565e-13
```

```
> library(ROC)
> truth <- c(rep(1, 200), rep(0, 800))
> result$pvalue[is.na(result$pvalue)] <- 1
> AUC(rocdemo.sca(truth = truth, data = -rank(result$pvalue)))
```

```
[1] 0.8632313
```

4.2 DE analysis for two-group data without replicates (paired)

edgeR and **DESeq2** employs generalized linear models (GLMs) which can apply to detect DEGs from paired two-group count data. The analysis for paired samples can be performed by indicating (1) the pair information when constructing the `TCC` class object and (2) `paired = TRUE` when performing the `calcNormFactors` and `estimateDE` functions. For analyzing paired data, we here use the hypothetical count data (`hypoData`; for details see 2.2) by changing the label information, i.e.,

```
> data(hypoData)
> colnames(hypoData) <- c("T_dogA", "T_dogB", "T_dogC",
+                         "N_dogA", "N_dogB", "N_dogC")
> head(hypoData)
```

```
       T_dogA T_dogB T_dogC N_dogA N_dogB N_dogC
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
```

This count data consists of three individuals (or sibships), dogA, dogB, and dogC. "T" and "N" indicate tumor and normal tissues, respectively.

We give a procedure for DE analysis using the likelihood ratio test for GLMs implemented in **edgeR** together with iDEGES/edgeR normalization factors (i.e., the iDEGES/edgeR-edgeR combination) for the paired two-group count data without replicates (i.e., the above `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)
> colnames(hypoData) <- c("T_dogA", "T_dogB", "T_dogC",
+                         "N_dogA", "N_dogB", "N_dogC")
> group <- c(1, 1, 1, 2, 2, 2)
> pair <- c(1, 2, 3, 1, 2, 3)
```

```
> cl <- data.frame(group = group, pair = pair)
> tcc <- new("TCC", hypoData, cl)
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 3, FDR = 0.1, floorPDEG = 0.05, paired = TRUE)
> tcc <- estimateDE(tcc, test.method = "edger", FDR = 0.1, paired = TRUE)
```

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
1 gene_151 9.736875 -2.731260 6.311487e-05 0.01588475    1            1
2  gene_68 6.208927 -2.844203 1.069249e-04 0.01588475    2            1
3  gene_11 8.772136 -2.082343 1.332242e-04 0.01588475    3            1
4  gene_16 7.348416 -2.314446 1.807105e-04 0.01588475    4            1
5 gene_144 7.588415 -2.107755 1.836923e-04 0.01588475    5            1
6 gene_105 9.350936 -1.971187 1.863223e-04 0.01588475    6            1
```

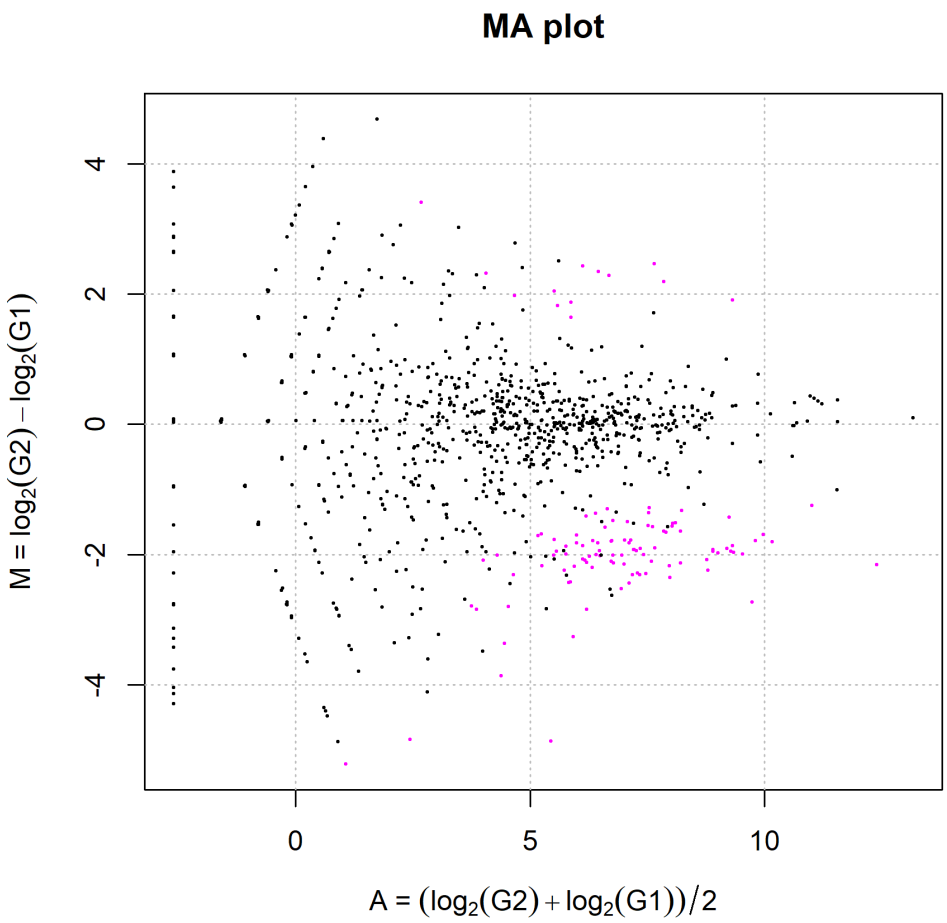
The DE results can be broken down as follows.

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

```
  0   1
889 111
```

This means 889 non-DEGs and 111 DEGs satisfy FDR < 0.1. The `plot` function generates an M-A plot, where "M" indicates the log-ratio (i.e., $\mathrm{M} = \log_2 \mathrm{G2} - \log_2 \mathrm{G1}$) and "A" indicates average read count (i.e., $\mathrm{A} = (\log_2 \mathrm{G2} + \log_2 \mathrm{G1})/2$), from the normalized count data. The magenta points indicate the identified DEGs at FDR < 0.1.

```
> plot(tcc)
```



Since we know the true information for `hypoData` (i.e., 200 DEGs and 800 non-DEGs), we can calculate the area under the ROC curve (i.e., AUC; $0 \leq \text{AUC} \leq 1$) between the ranked gene list and the truth 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).

```
> library(ROC)
> truth <- c(rep(1, 200), rep(0, 800))
> AUC(rocdemo.sca(truth = truth, data = -tcc$stat$rank))
```

```
[1] 0.8685719
```

For comparison, the DE results from the original procedure in **edgeR** can be obtained as follows.

```
> library(TCC)
> data(hypoData)
> colnames(hypoData) <- c("T_dogA", "T_dogB", "T_dogC",
+                         "N_dogA", "N_dogB", "N_dogC")
> group <- factor(c(1, 1, 1, 2, 2, 2))
> pair <- factor(c(1, 2, 3, 1, 2, 3))
> design <- model.matrix(~ group + pair)
> coef <- 2
> d <- DGEList(counts = hypoData, group = group)
> d <- calcNormFactors(d)
> d <- estimateDisp(d, design)
> fit <- glmQLFit(d, design)
> out <- glmQLFTest(fit, coef=2)
> topTags(out, n = 6)
```

```
Coefficient:  group2
             logFC    logCPM        F       PValue        FDR
gene_151 -2.572244 13.268838 92.30316 6.010600e-05 0.01694428
gene_68  -2.767267  9.808735 80.28985 9.022718e-05 0.01694428
gene_189  2.538559  9.661587 73.77578 1.152545e-04 0.01694428
gene_196  2.506855  9.964259 73.47536 1.166192e-04 0.01694428
gene_11  -1.953193 12.094808 66.59564 1.547045e-04 0.01694428
gene_16  -2.305265 10.748393 66.51179 1.552638e-04 0.01694428
```

```
> p.value <- out$table$PValue
> library(ROC)
> truth <- c(rep(1, 200), rep(0, 800))
> AUC(rocdemo.sca(truth = truth, data = -rank(p.value)))
```

```
[1] 0.8518594
```

This is the same as

```
> library(TCC)
> data(hypoData)
> colnames(hypoData) <- c("T_dogA", "T_dogB", "T_dogC",
+                         "N_dogA", "N_dogB", "N_dogC")
> group <- c(1, 1, 1, 2, 2, 2)
> pair <- c(1, 2, 3, 1, 2, 3)
> cl <- data.frame(group = group, pair = pair)
> tcc <- new("TCC", hypoData, cl)
> tcc <- calcNormFactors(tcc, norm.method = "tmm", iteration = 0, paired = TRUE)
> tcc <- estimateDE(tcc, test.method = "edger", FDR = 0.1, paired = TRUE)
> result <- getResult(tcc, sort = TRUE)
> head(result)
```

```
   gene_id  a.value   m.value      p.value    q.value rank estimatedDEG
1 gene_151 9.740529 -2.600366 6.010600e-05 0.01694428    1            1
2  gene_68 6.209636 -2.723799 9.022718e-05 0.01694428    2            1
3 gene_189 6.127498  2.550316 1.152545e-04 0.01694428    3            1
4 gene_196 6.461462  2.475704 1.166192e-04 0.01694428    4            1
5  gene_11 8.774575 -1.957134 1.547045e-04 0.01694428    5            1
6  gene_16 7.346399 -2.192904 1.552638e-04 0.01694428    6            1
```

```
> library(ROC)
> truth <- c(rep(1, 200), rep(0, 800))
> AUC(rocdemo.sca(truth = truth, data = -tcc$stat$rank))
```

```
[1] 0.8518594
```

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  ###
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 1)
> ###  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
1  gene_52      NA      NA 9.873844e-09 9.873844e-09    1            1
2 gene_179      NA      NA 2.051493e-08 1.025747e-08    2            1
3 gene_121      NA      NA 6.761494e-08 2.253831e-08    3            1
4  gene_88      NA      NA 1.221338e-07 3.053346e-08    4            1
5  gene_64      NA      NA 2.483747e-07 4.967494e-08    5            1
6 gene_140      NA      NA 5.728008e-07 9.546681e-08    6            1
```

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

```
  0   1
869 131
```

It can be seen that the **baySeq** method identified 131 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] 159
```

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  ###
> tcc <- calcNormFactors(tcc, norm.method = "tmm", test.method = "edger",
+                        iteration = 1)
> ###  DE analysis  ###
> 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
1 gene_121      NA      NA 2.659198e-07 0.0001054174    1            1
2  gene_64      NA      NA 3.463828e-07 0.0001054174    2            1
3 gene_134      NA      NA 6.382694e-07 0.0001054174    3            1
4  gene_74      NA      NA 7.898076e-07 0.0001054174    4            1
5 gene_140      NA      NA 8.085682e-07 0.0001054174    5            1
6  gene_88      NA      NA 8.163473e-07 0.0001054174    6            1
```

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

```
  0   1
880 120
```

5 Generation of simulation data

5.1 Introduction and basic usage

As demonstrated in our previous study (Kadota et al., 2012), 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). 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), 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. 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     347      65     267       3      14      57
gene_2      10     114      87       4       4       3
gene_3     121      44      84      15      17      13
gene_4      62      18       7       1      19      12
gene_5       2       4      11       1       3       5
gene_6     353     338     212      59      50      77
```

```
> 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).

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 4
 $ trueDEG       : Named num [1:1000] 1 1 1 1 1 1 1 1 1 1 ...
  ..- attr(*, "names")= chr [1:1000] "gene_1" "gene_2" "gene_3" "gene_4" ...
 $ DEG.foldchange: num [1:1000, 1:6] 4 4 4 4 4 4 4 4 4 4 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:1000] "gene_1" "gene_2" "gene_3" "gene_4" ...
  .. ..$ : chr [1:6] "G1_rep1" "G1_rep2" "G1_rep3" "G2_rep1" ...
 $ PDEG          : num [1:2] 0.18 0.02
 $ params        :'data.frame':        1000 obs. of  2 variables:
  ..$ mean: num [1:1000] 30.82 14.06 16.98 12.97 1.94 ...
  ..$ disp: num [1:1000] 0.8363 1.8306 0.0962 0.5087 0.5527 ...
```

Specifically, the entries for 0 and 1 in the `tcc$simulation$trueDEG` object are for non-DEGs and DEGs respectively. The breakdowns for individual entries are the same as stated above: 800 entries are non-DEGs, 200 entries are DEGs.

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

```
  0   1
800 200
```

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) 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
1  gene_63 10.935582 -2.149818 1.847395e-06 0.001847395    1            1
2 gene_170 10.192448 -1.927699 4.038285e-06 0.001852539    2            1
3  gene_83 12.301057 -1.874640 8.153956e-06 0.001852539    3            1
4  gene_57 11.525795 -1.955876 8.555850e-06 0.001852539    4            1
5 gene_190  8.467597  1.977314 1.387586e-05 0.001852539    5            1
6  gene_23  8.702161 -2.116659 1.599079e-05 0.001852539    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] 0.906825
```

This is essentially the same as

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

```
[1] 0.906825
```

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] 0.8907625
```

The following is an alternative procedure for **edgeR** users.

```
> design <- model.matrix(~ as.factor(tcc$group$group))
> d <- DGEList(counts = tcc$count, group = tcc$group$group)
> d <- calcNormFactors(d)
> d$samples$norm.factors <- d$samples$norm.factors /
+                           mean(d$samples$norm.factors)
> d <- estimateDisp(d, design)
> fit <- glmQLFit(d, design)
> result <- glmQLFTest(fit, coef = 2)
> result$table$PValue[is.na(result$table$PValue)] <- 1
> AUC(rocdemo.sca(truth = as.numeric(tcc$simulation$trueDEG != 0),
+                 data = -rank(result$table$PValue)))
```

```
[1] 0.8907625
```

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 (0.9068250) of DEGES/edgeR is quite similar to the AUC value (0.9077000) 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] 0.9077
```

5.2 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
```

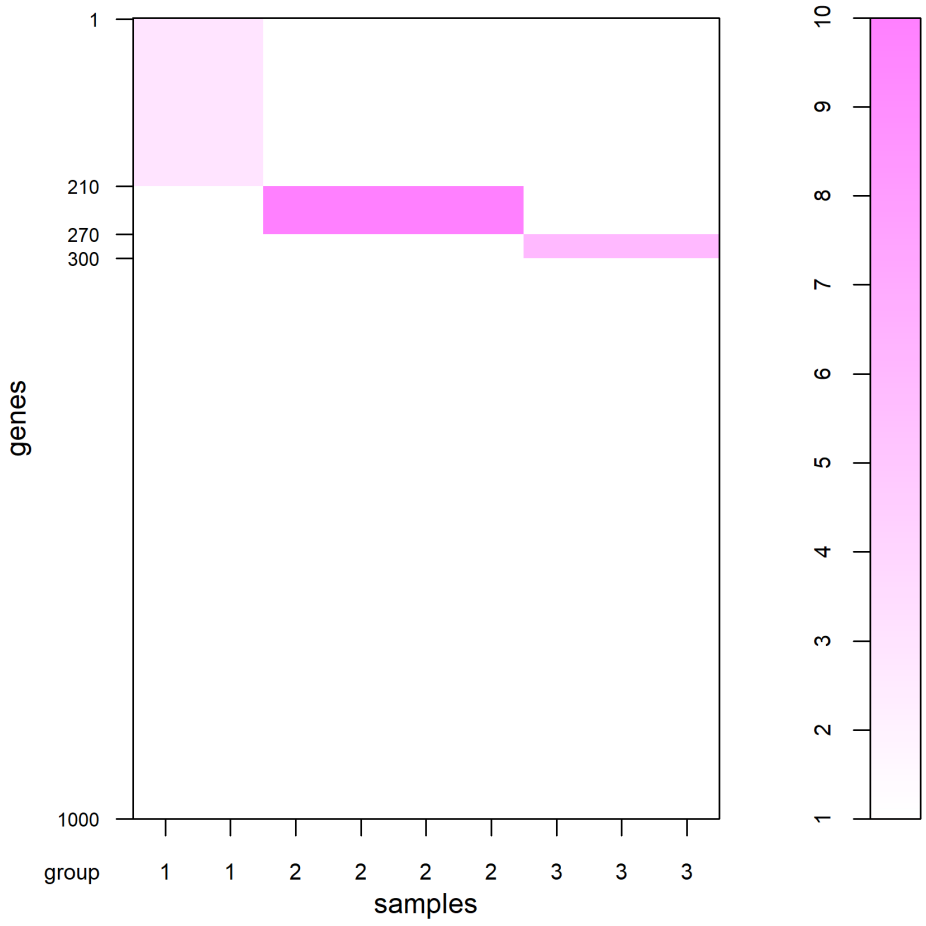
```
        group
G1_rep1     1
G1_rep2     1
G2_rep1     2
G2_rep2     2
G2_rep3     2
G2_rep4     2
G3_rep1     3
G3_rep2     3
G3_rep3     3
```

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

```
       G1_rep1 G1_rep2 G2_rep1 G2_rep2 G2_rep3 G2_rep4 G3_rep1 G3_rep2 G3_rep3
gene_1     259      63      37       5      24      12       4      66       3
gene_2       8      12       2      13      47       2       0      22      24
gene_3      92      64      23      14      18      18      23      20      20
gene_4      48      43      14      24       9      12       6       8       5
gene_5       2       6       4       3       2       0       7       5       0
gene_6     264     237      81      82      73      60      38      87     108
```

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)
```



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

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

```
[1] 0.8605143
```

Next, we calculate the corresponding value using the original **edgeR** procedure for single factor experimental design (i.e., the edgeR-edgeR combination).

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

```
[1] 0.8505
```

It can be seen that the DEGES/edgeR-edgeR combination outperforms the original **edgeR** procedure under the given simulation conditions. Note that the `test.method` argument will be ignored when `iteration = 0` is specified.

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
```

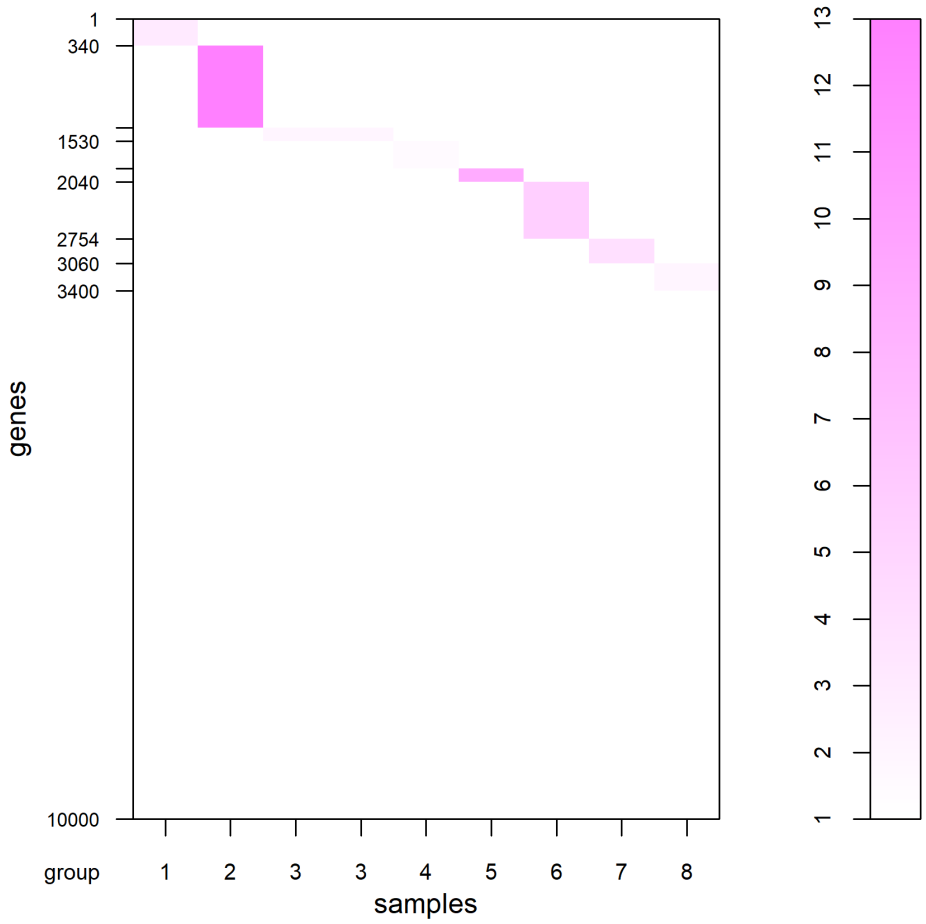
```
> tcc$group
```

```
        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      16      16      14      63      16      62      64      42       9
gene_2       2       0      24      10       3      22      14      40       3
gene_3      72      27      25      17      20      20      10      20      12
gene_4      32      16       6       7      18       6       1       0      22
gene_5       0       1       6       0       2       0       4       1       8
gene_6     327     182     172     158     107      87      72     147      76
```

```
> plotFCPseudocolor(tcc)
```



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

5.3 Multi-factor data

The `simulateReadCounts` function can also generate simulation data in multi-factor experimental design. Different from above single-factor experimental design, the `group` argument should be used instead of `replicates` for specifying sample conditions (or factors) when generating simulation data in multi-factor design. In relation to the `group` specification, the `DEG.foldchange` argument should also be specified as a data frame object.

We generate a dataset consisting of two factors for comparing (i) two Groups (i.e., "WT" vs. "KO") as the first factor, at (ii) two time points (i.e., "1d" vs. "2d") as the second factor, with all samples obtained from independent subjects. There are a total of four conditions ("WT_1d", "WT_2d", "KO_1d", and "KO_2d") each of which has two biological replicates, comprising a total of eight samples. The `group` argument for this experimental design can be described as follows:

```
> group <- data.frame(
+     GROUP = c("WT", "WT", "WT", "WT", "KO", "KO", "KO", "KO"),
+     TIME = c("1d", "1d", "2d", "2d", "1d", "1d", "2d", "2d")
+ )
```

Next, we design the number of types of DEGs and the levels of fold-change by the `DEG.foldchange` argument. We here introduce three types of DEGs: (a) 2-fold up-regulation in the first four samples (i.e., "WT"), (b) 3-fold up-regulation in the last four samples (i.e., "KO"), and (c) 2-fold down-regulation at "2d" in "WT" and 4-fold up-regulation at "2d" in "KO". This implies that the first two types of DEGs are related to the first factor (i.e., "WT" vs. "KO") and the third type of DEG is related to the second factor (i.e., "1d" vs. "2d").

```
> DEG.foldchange <- data.frame(
+     FACTOR1.1 = c(2, 2, 2, 2, 1, 1, 1, 1),
+     FACTOR1.2 = c(1, 1, 1, 1, 3, 3, 3, 3),
+     FACTOR2 = c(1, 1, 0.5, 0.5, 1, 1, 4, 4)
+ )
```

The other simulation conditions for this dataset are as follows: (1) the number of gene is 1,000 (i.e., `Ngene = 1000`), (2) the first 20% of genes are DEGs (i.e., `PDEG = 0.2`), and (3) the breakdowns of the three types of DEGs are 50%, 20%, and 30% (i.e., `DEG.assign = c(0.5, 0.2, 0.3)`).

```
> set.seed(1000)
> tcc <- simulateReadCounts(Ngene = 10000, PDEG = 0.2,
+                 DEG.assign = c(0.5, 0.2, 0.3),
+                 DEG.foldchange = DEG.foldchange,
+                 group = group)
```

Since the first six rows in the dataset corresponds to the first type of DEGs, we can see the 2-fold up-regulation in the first four columns (i.e., WT-related samples) compared to the last four columns (i.e., KO-related samples).

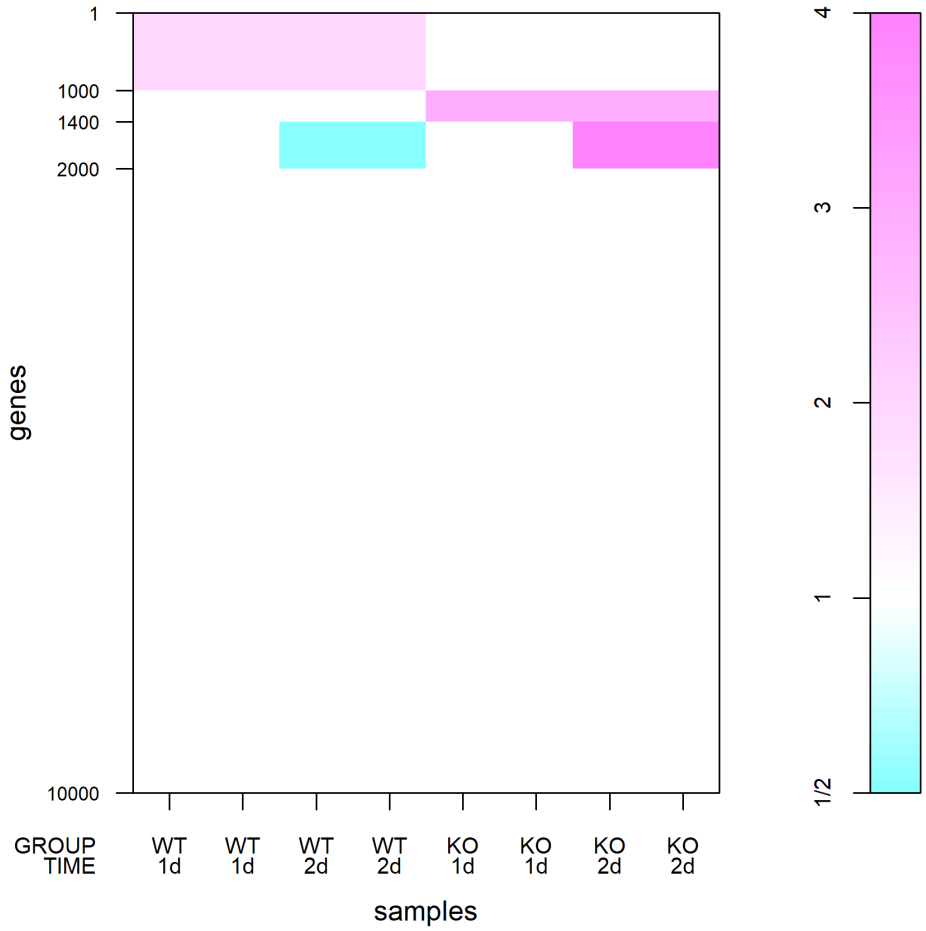
```
> head(tcc$count)
```

```
       WT1d_rep1 WT1d_rep2 WT2d_rep1 WT2d_rep2 KO1d_rep1 KO1d_rep2 KO2d_rep1
gene_1        12        33        14       130        39         4        40
gene_2        14        23         1        19        13         2         2
gene_3        21        64        26        27        11        16        10
gene_4        12        36        47        10         8        15        16
gene_5         2         0         1         2         1         4         0
gene_6       174       110       197       115        58        30        69
       KO2d_rep2
gene_1        42
gene_2        40
gene_3        20
gene_4         0
gene_5         1
gene_6       147
```

```
> tcc$group
```

```
          GROUP TIME
WT1d_rep1    WT   1d
WT1d_rep2    WT   1d
WT2d_rep1    WT   2d
WT2d_rep2    WT   2d
KO1d_rep1    KO   1d
KO1d_rep2    KO   1d
KO2d_rep1    KO   2d
KO2d_rep2    KO   2d
```

```
> plotFCPseudocolor(tcc)
```



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)`).

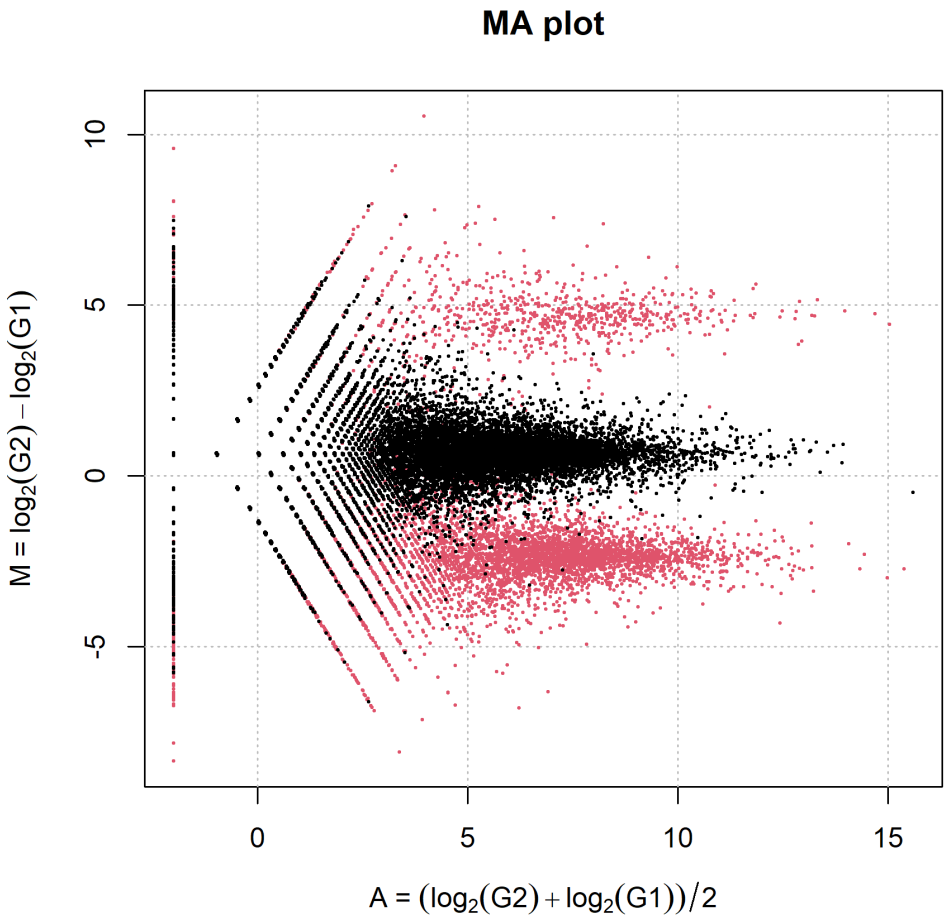
```
> set.seed(1000)
> library(TCC)
> tcc <- simulateReadCounts(Ngene = 20000, PDEG = 0.30,
+                 DEG.assign = c(0.85, 0.15),
+                 DEG.foldchange = c(8, 16),
+                 replicates = c(2, 2))
> head(tcc$count)
```

```
       G1_rep1 G1_rep2 G2_rep1 G2_rep2
gene_1     415     238       0     140
```

```
gene_2       0       73       0       9
gene_3     148      110      10      31
gene_4     134      189      28       3
gene_5      32       18       1       0
gene_6     467      363      93      53
```

An M-A plot for the simulation data can be viewed as follows; the points for DEGs are colored red and the non-DEGs are colored 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
4155848 4155848 4155848 4155848
```

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

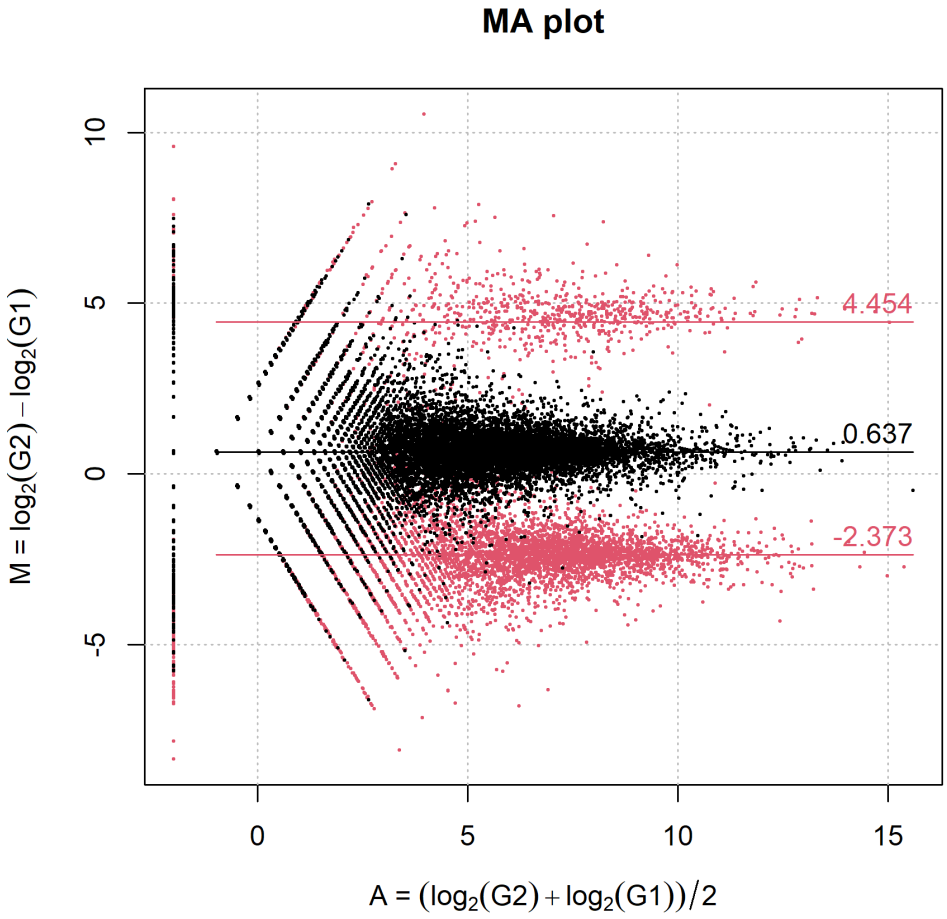
```
G1_rep1 G1_rep2 G2_rep1 G2_rep2
4954709 5152169 3226887 3289627
```

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

```
[1] 4155848
```

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 0.637, −2.373, and 4.454 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 (= 0.070) 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 4.1.1 (2021-08-10)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows Server x64 (build 17763)

Matrix products: default

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

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

other attached packages:
 [1] TCC_1.34.0                  ROC_1.70.0
 [3] baySeq_2.28.0               abind_1.4-5
 [5] edgeR_3.36.0                limma_3.50.0
 [7] DESeq2_1.34.0               SummarizedExperiment_1.24.0
 [9] Biobase_2.54.0              MatrixGenerics_1.6.0
[11] matrixStats_0.61.0          GenomicRanges_1.46.0
[13] GenomeInfoDb_1.30.0         IRanges_2.28.0
[15] S4Vectors_0.32.0            BiocGenerics_0.40.0

loaded via a namespace (and not attached):
 [1] locfit_1.5-9.4         Rcpp_1.0.7             lattice_0.20-45
 [4] png_0.1-7              Biostrings_2.62.0      assertthat_0.2.1
 [7] utf8_1.2.2             R6_2.5.1               RSQLite_2.2.8
[10] httr_1.4.2             ggplot2_3.3.5          pillar_1.6.4
[13] zlibbioc_1.40.0        rlang_0.4.12           rstudioapi_0.13
[16] annotate_1.72.0        blob_1.2.2             Matrix_1.3-4
[19] splines_4.1.1          BiocParallel_1.28.0    geneplotter_1.72.0
[22] RCurl_1.98-1.5         bit_4.0.4              munsell_0.5.0
[25] DelayedArray_0.20.0    xfun_0.27              compiler_4.1.1
[28] pkgconfig_2.0.3        tidyselect_1.1.1       KEGGREST_1.34.0
[31] tibble_3.1.5           GenomeInfoDbData_1.2.7 XML_3.99-0.8
[34] fansi_0.5.0            crayon_1.4.1           dplyr_1.0.7
[37] bitops_1.0-7           grid_4.1.1             xtable_1.8-4
[40] gtable_0.3.0           lifecycle_1.0.1        DBI_1.1.1
[43] magrittr_2.0.1         scales_1.1.1           cachem_1.0.6
[46] XVector_0.34.0         genefilter_1.76.0      ellipsis_0.3.2
[49] vctrs_0.3.8            generics_0.1.1         RColorBrewer_1.1-2
[52] tools_4.1.1            bit64_4.0.5            glue_1.4.2
[55] purrr_0.3.4            fastmap_1.1.0          survival_3.2-13
[58] AnnotationDbi_1.56.0   colorspace_2.0-2       memoise_2.0.0
[61] knitr_1.36
```

7 References

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

[2] Love MI, Huber W, Anders S. 2014. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol 15: R550.

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

[4] 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; French StatOmique Consortium. 2013. A comprehensive evaluation of normalization methods for Illumina high-throughput RNA sequencing data analysis. Brief Bioinform 14: 671-683.

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

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

[7] Kadota K, Nakai Y, Shimizu K. 2008. A weighted average difference method for detecting differentially expressed genes from microarray data. Algorithms Mol Biol 3: 8.

[8] Kadota K, Nishimura SI, Bono H, Nakamura S, Hayashizaki Y, Okazaki Y, Takahashi K. 2003. Detection of genes with tissue-specific expression patterns using Akaike's Information Criterion (AIC) procedure. Physiol Genomics 12: 251-259.

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

[10] Kadota K, Ye J, Nakai Y, Terada T, Shimizu K. 2006. ROKU: a novel method for identification of tissue-specific genes. BMC Bioinformatics 7: 294.

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

[12] Robinson MD, McCarthy DJ, Smyth GK. 2010. edgeR: A Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26: 139-140.

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

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

[15] Sun J, Nishiyama T, Shimizu K, and Kadota K. TCC: an R package for comparing tag count data with robust normalization strategies. BMC Bioinformatics 14: 219.

[16] Ueda T. 1996. Simple method for the detection of outliers. Japanese J Appl Stat 25: 17-26.