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). 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
  1.1 Installation .................................................... 3
  1.2 Citations ....................................................... 4
  1.3 Quick start ....................................................... 4

### 2 Preparations
  2.1 Preparing the count data ........................................ 5
  2.2 Reading the count data .......................................... 5
  2.3 Constructing TCC class object ................................... 6
  2.4 Filtering low-count genes (optional) ..................... 7

### 3 Normalization
  3.1 Normalization of two-group count data with replicates ........ 9
    3.1.1 DEGES/edgeR ............................................. 9
    3.1.2 iDEGES/edgeR ........................................... 10
    3.1.3 DEGES/DESeq2 ........................................... 11
  3.2 Normalization of two-group count data without replicates (paired) ... 12
    3.2.1 DEGES/edgeR ............................................. 12
  3.3 Normalization of multi-group count data with replicates ........ 13
    3.3.1 DEGES/edgeR ............................................. 14
  3.4 Retrieving normalized data .................................... 16
    3.4.1 Retrieving two-group DEGES/edgeR-normalized data with replicates ... 17
    3.4.2 Retrieving two-group DEGES/edgeR-normalized data without replicates (paired) ... 18
    3.4.3 Retrieving multi-group iDEGES/edgeR-normalized data with replicates ... 19

### 4 Differential expression (DE)
  4.1 DE analysis for two-group data with replicates ................ 21
    4.1.1 edgeR coupled with iDEGES/edgeR normalization ................ 21
    4.1.2 DESeq2 coupled with iDEGES/DESeq2 normalization ....... 24
  4.2 DE analysis for two-group data without replicates (paired) .... 27
  4.3 DE analysis for multi-group data with replicates ............... 30
    4.3.1 edgeR coupled with DEGES/edgeR normalization .......... 31

### 5 Generation of simulation data .................................. 31
  5.1 Introduction and basic usage .................................. 31
  5.2 Multi-group data with and without replicates ................. 34
  5.3 Multi-factor data .............................................. 37
  5.4 Other utilities .................................................. 39

### 6 Session info .................................................... 43

### 7 References ...................................................... 44
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 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), and DESeq (Anders and Huber, 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 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:
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 an 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 3.1.2 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)

<table>
<thead>
<tr>
<th>gene_id</th>
<th>a.value</th>
<th>m.value</th>
<th>p.value</th>
<th>q.value</th>
<th>rank</th>
<th>estimatedDEG</th>
</tr>
</thead>
<tbody>
<tr>
<td>gene_144</td>
<td>7.588245</td>
<td>-2.124465</td>
<td>7.250541e-06</td>
<td>0.002087657</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>gene_168</td>
<td>8.903727</td>
<td>-1.963527</td>
<td>9.930405e-06</td>
<td>0.002087657</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>gene_123</td>
<td>8.213471</td>
<td>-2.145537</td>
<td>1.340805e-05</td>
<td>0.002087657</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>gene_39</td>
<td>8.113471</td>
<td>-2.145347</td>
<td>1.340805e-05</td>
<td>0.002087657</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>gene_11</td>
<td>8.722684</td>
<td>-2.096001</td>
<td>1.474953e-05</td>
<td>0.002087657</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>gene_151</td>
<td>9.735594</td>
<td>-2.745972</td>
<td>1.477748e-05</td>
<td>0.002087657</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>
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.

```r
> 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     44     18     0      0      0
gene_5      98     48     17     1      8      5
gene_6   296  282   216     86     62     69
```

```r
> dim(hypoData)
[1] 1000   6
```

```r
> 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.

```r
> group <- c(1, 1, 1, 2, 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.

```r
> 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.

```r
> 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
```

```r
> 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.

```r
> 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.

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

[1] 996 6

```r
> 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 and DESeq2) 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>
<thead>
<tr>
<th>Table 1: 3.1 unpaired samples (two-group)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>G1_rep1</td>
</tr>
<tr>
<td>G1_rep2</td>
</tr>
<tr>
<td>G1_rep3</td>
</tr>
<tr>
<td>G2_rep1</td>
</tr>
<tr>
<td>G2_rep2</td>
</tr>
<tr>
<td>G2_rep3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2: 3.2 paired samples (two-group)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>G1_rep1</td>
</tr>
<tr>
<td>G1_rep2</td>
</tr>
<tr>
<td>G1_rep3</td>
</tr>
<tr>
<td>G2_rep1</td>
</tr>
<tr>
<td>G2_rep2</td>
</tr>
<tr>
<td>G2_rep3</td>
</tr>
</tbody>
</table>
Table 3: 3.3 unpaired samples (multi-group)

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

### 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 present three shorter alternatives (3.1.1 DEGES/edgeR, 3.1.2 iDEGES/edgeR, and 3.1.3 DEGES/DESeq2).

#### 3.1.1 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 \( P_{\text{DEG}} \). The default FDR is \(< 0.1\), and the default floor \( P_{\text{DEG}} \) 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:

```r
> 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
```

<table>
<thead>
<tr>
<th>G1_rep1</th>
<th>G1_rep2</th>
<th>G1_rep3</th>
<th>G2_rep1</th>
<th>G2_rep2</th>
<th>G2_rep3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8780077</td>
<td>0.8602327</td>
<td>0.8468366</td>
<td>1.0848669</td>
<td>1.1463246</td>
<td>1.1837315</td>
</tr>
</tbody>
</table>
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.

```r
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)
out <- glmQLTest(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)
```

### 3.1.2 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:

```r
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
```

<table>
<thead>
<tr>
<th>G1_rep1</th>
<th>G1_rep2</th>
<th>G1_rep3</th>
<th>G2_rep1</th>
<th>G2_rep2</th>
<th>G2_rep3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.878008</td>
<td>0.860233</td>
<td>0.846837</td>
<td>1.084867</td>
<td>1.146324</td>
<td>1.183731</td>
</tr>
</tbody>
</table>
3.1.3 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:

```r
> 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
```

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.

```r
> 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 × two DEG identification methods) coupled with an arbitrary number of iterations \((n = 0, 1, 2, \ldots, 100)\) 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   368    388     22      26      25     68
gene_3  1144    919     990     374     480     239
gene_4    0      0      44       0       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)
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.

```r
> 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
0.8787604 0.8621212 0.8466855 1.0844390 1.1444736 1.1835203
```

3.3 Normalization of multi-group count data with replicates

Many R packages (including edgeR) 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:

<table>
<thead>
<tr>
<th>gene_1</th>
<th>G1_rep1</th>
<th>G1_rep2</th>
<th>G1_rep3</th>
<th>G2_rep1</th>
<th>G2_rep2</th>
<th>G2_rep3</th>
<th>G3_rep1</th>
<th>G3_rep2</th>
<th>G3_rep3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>63</td>
<td>48</td>
<td>31</td>
<td>15</td>
<td>12</td>
<td>12</td>
<td>24</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>gene_2</td>
<td>18</td>
<td>0</td>
<td>7</td>
<td>2</td>
<td>3</td>
<td>8</td>
<td>3</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>gene_3</td>
<td>106</td>
<td>66</td>
<td>25</td>
<td>9</td>
<td>14</td>
<td>14</td>
<td>11</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>gene_4</td>
<td>4</td>
<td>9</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>gene_5</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>gene_6</td>
<td>57</td>
<td>100</td>
<td>83</td>
<td>20</td>
<td>5</td>
<td>16</td>
<td>26</td>
<td>7</td>
<td>21</td>
</tr>
</tbody>
</table>

Sample:

| group norm.factors lib.sizes |
|-----------------------------|------------------------|
| G1_rep1                     | 1                      | 150490 |
| G1_rep2                     | 1                      | 166665 |
| G1_rep3                     | 1                      | 199283 |
| G2_rep1                     | 2                      | 183116 |
| G2_rep2                     | 2                      | 126651 |
| G2_rep3                     | 2                      | 131377 |
| G3_rep1                     | 3                      | 149828 |
| G3_rep2                     | 3                      | 150288 |
| G3_rep3                     | 3                      | 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 a DEGES-based normalization pipeline for this multi-group data here.

### 3.3.1 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$n.norm.factors

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>G1_rep1</td>
<td>G1_rep2</td>
<td>G1_rep3</td>
<td>G2_rep1</td>
<td>G2_rep2</td>
<td>G2_rep3</td>
<td>G3_rep1</td>
<td>G3_rep2</td>
</tr>
<tr>
<td>1.0200385</td>
<td>0.9122158</td>
<td>0.7823038</td>
<td>0.8320162</td>
<td>1.1749457</td>
<td>1.2023022</td>
<td>1.0210889</td>
<td>1.0285490</td>
</tr>
<tr>
<td>G3_rep3</td>
<td>1.0265398</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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$n.norm.factors

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>G1_rep1</td>
<td>G1_rep2</td>
<td>G1_rep3</td>
<td>G2_rep1</td>
<td>G2_rep2</td>
<td>G2_rep3</td>
<td>G3_rep1</td>
<td>G3_rep2</td>
</tr>
<tr>
<td>1.0200385</td>
<td>0.9122158</td>
<td>0.7823038</td>
<td>0.8320162</td>
<td>1.1749457</td>
<td>1.2023022</td>
<td>1.0210889</td>
<td>1.0285490</td>
</tr>
<tr>
<td>G3_rep3</td>
<td>1.0265398</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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$n.norm.factors * colSums(hypoData_mg[!is.DEG, ])/
+    colSums(hypoData_mg)
> norm.factors <- norm.factors / mean(norm.factors)
> norm.factors
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$normFactors`, 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 Tt paper (Kadota et al., 2012); i.e., (i) the first 200 genes are DEGs ($P_{DEG} = 200/1000 = 20\%$), (ii) the first 180 genes of the 200 DEGs are higher in G1 ($P_{G1} = 180/200 = 90\%$), 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.

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

        G1_rep1      G1_rep2      G1_rep3      G2_rep1
Min. :  0.00  Min. :  0.00  Min. :  0.00  Min. :  0.00
1st Qu. :  3.00  1st Qu. :  3.00  1st Qu. :  3.00  1st Qu. :  3.00
Median : 20.50  Median : 20.00  Median : 20.00  Median : 21.00
Mean : 103.36  Mean : 104.45  Mean : 104.45  Mean : 113.80
3rd Qu. : 74.25  3rd Qu. : 73.25  3rd Qu. : 73.25  3rd Qu. : 68.00
Max. : 8815.00  Max. : 9548.00  Max. : 9548.00  Max. : 9304.00

        G2_rep2      G2_rep3
Min. :  0.00  Min. :  0.00
1st Qu. :  3.00  1st Qu. :  3.00
Median :  21.00  Median :  20.00
Mean : 105.00  Mean : 104.60
3rd Qu. :  70.00  3rd Qu. :  70.00
Max. : 9466.00  Max. : 9320.00

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

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

```r
> 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.

```r
> 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, , , median]
```

<table>
<thead>
<tr>
<th></th>
<th>G1_rep1</th>
<th>G1_rep2</th>
<th>G1_rep3</th>
<th>G2_rep1</th>
<th>G2_rep2</th>
<th>G2_rep3</th>
</tr>
</thead>
</table>

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


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.

```r
> 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, , , median)
```

<table>
<thead>
<tr>
<th></th>
<th>G1_rep1</th>
<th>G1_rep2</th>
<th>G1_rep3</th>
<th>G2_rep1</th>
<th>G2_rep2</th>
<th>G2_rep3</th>
</tr>
</thead>
</table>

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

[1] 18.59060 22.98591

It is obvious that the summary statistics (ranging from 19.39166 to 21.65596 ) 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.

```r
> 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   33.495  44.2196  118.3469  16.1999  14.4625  29.5754
gene_2   352.684  381.2704  21.3374  36.4497  25.8259  69.3493
gene_3  1127.012  903.0605  960.1839  378.6720  495.8587  243.7423
gene_4    0.0000  0.0000  42.6748  18.2249  0.0000  0.0000

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

T_dogA  T_dogB  T_dogC  N_dogA  N_dogB  N_dogC
------  -------  -------  ------  ------  ------

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


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.

```r
> 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
------  -------  -------  ------  ------  ------

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

[1] 18.59060 22.98591

It is obvious that the summary statistics (ranging from 19.39765 to 21.69382) 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.

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

<table>
<thead>
<tr>
<th></th>
<th>G1_rep1</th>
<th>G1_rep2</th>
<th>G1_rep3</th>
<th>G2_rep1</th>
<th>G2_rep2</th>
<th>G2_rep3</th>
<th>G3_rep1</th>
<th>G3_rep2</th>
<th>G3_rep3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>0.00</td>
<td>0.00</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>2.00</td>
<td>1st Qu.</td>
<td>2.0</td>
<td>1st Qu.</td>
<td>2.0</td>
<td>1st Qu.</td>
<td>2.0</td>
<td>1st Qu.</td>
<td>2.0</td>
</tr>
<tr>
<td>Median</td>
<td>14.00</td>
<td>Median</td>
<td>13.0</td>
<td>Median</td>
<td>14.5</td>
<td>Median</td>
<td>13.0</td>
<td>Median</td>
<td>13.0</td>
</tr>
<tr>
<td>Mean</td>
<td>133.41</td>
<td>Mean</td>
<td>150.5</td>
<td>Mean</td>
<td>190.6</td>
<td>Mean</td>
<td>199.4</td>
<td>Mean</td>
<td>52.0</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>51.25</td>
<td>3rd Qu.</td>
<td>53.0</td>
<td>3rd Qu.</td>
<td>55.0</td>
<td>3rd Qu.</td>
<td>52.0</td>
<td>3rd Qu.</td>
<td>52.0</td>
</tr>
<tr>
<td>Max.</td>
<td>27218.00</td>
<td>Max.</td>
<td>27987.0</td>
<td>Max.</td>
<td>66273.0</td>
<td>Max.</td>
<td>75148.0</td>
<td>Max.</td>
<td>75148.0</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th></th>
<th>G1_rep1</th>
<th>G1_rep2</th>
<th>G1_rep3</th>
<th>G2_rep1</th>
<th>G2_rep2</th>
<th>G2_rep3</th>
<th>G3_rep1</th>
<th>G3_rep2</th>
<th>G3_rep3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>14.0</td>
<td>13.0</td>
<td>14.5</td>
<td>13.0</td>
<td>13.0</td>
<td>14.0</td>
<td>14.0</td>
<td>15.0</td>
<td>15.0</td>
</tr>
</tbody>
</table>

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

```r
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)
```

<table>
<thead>
<tr>
<th></th>
<th>G1_rep1</th>
<th>G1_rep2</th>
<th>G1_rep3</th>
<th>G2_rep1</th>
<th>G2_rep2</th>
<th>G2_rep3</th>
<th>G3_rep1</th>
<th>G3_rep2</th>
<th>G3_rep3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15.61819</td>
<td>15.0</td>
<td>15.0</td>
<td>15.0</td>
<td>15.0</td>
<td>15.0</td>
<td>15.0</td>
<td>15.0</td>
<td>15.0</td>
</tr>
</tbody>
</table>
For comparison, the summary statistics for TMM-normalized data produced using the original normalization method (i.e., TMM) in `edgeR` are obtained as follows.

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

[1] 12.98177 15.61819
```

It is obvious that the summary statistics (ranging from 12.98177 to 15.61819) 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 can be performed by using the `estimateDE` function with `test.method = "edger"`, and "deseq2", 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.

```r
> 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.

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

<table>
<thead>
<tr>
<th>gene_id</th>
<th>a.value</th>
<th>m.value</th>
<th>p.value</th>
<th>q.value</th>
<th>rank</th>
<th>estimatedDEG</th>
</tr>
</thead>
<tbody>
<tr>
<td>gene144</td>
<td>7.588245</td>
<td>-2.124465</td>
<td>7.250541e-06</td>
<td>0.002087657</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>gene168</td>
<td>8.903727</td>
<td>-1.963527</td>
<td>9.930405e-06</td>
<td>0.002087657</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>gene123</td>
<td>8.213471</td>
<td>-2.145537</td>
<td>1.340805e-05</td>
<td>0.002087657</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>gene39</td>
<td>7.110211</td>
<td>-2.099601</td>
<td>1.477953e-05</td>
<td>0.002087657</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>gene11</td>
<td>8.772624</td>
<td>-2.453971</td>
<td>1.477953e-05</td>
<td>0.002087657</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>gene151</td>
<td>9.735594</td>
<td>-2.745972</td>
<td>1.477953e-05</td>
<td>0.002087657</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

The DE results can be broken down as follows.

```r
> table(tcc$estimatedDEG)

0 1
869 131
```

This means 869 non-DEGs and 131 DEGs satisfy FDR < 0.1. The `plot` function generates an M-A plot, where "M" indicates the log-ratio (i.e., $M = \log_2 G_2 - \log_2 G_1$) and "A" indicates average read count (i.e., $A = (\log_2 G_2 + \log_2 G_1)/2$), from the normalized count data. The magenta points indicate the identified DEGs at FDR < 0.1.
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).

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

```
[1] 0.8878094
```

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

```r
> 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)
```
This is the same as

```r
> 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 1.230585e-05 0.002861706 1     1
2  gene_168 8.905896 -1.980620 1.230585e-05 0.002861706 2     1
3  gene_189 6.127498  2.550316 1.520716e-05 0.002861706 3     1
4  gene_11  8.774675 -1.957194 2.131080e-05 0.002861706 4     1
5  gene_39  7.114394 -2.308040 2.455803e-05 0.002861706 6     1
```

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

```
[1] 0.8690469
```

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.

```r
> 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 1.230585e-05 0.002861706 1     1
2  gene_168 8.905896 -1.980620 1.230585e-05 0.002861706 2     1
3  gene_189 6.127498  2.550316 1.520716e-05 0.002861706 3     1
4  gene_11  8.774675 -1.957194 2.131080e-05 0.002861706 4     1
5  gene_39  7.114394 -2.308040 2.455803e-05 0.002861706 6     1
```

The DE results can be broken down as follows.

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

```
  0  1
880 120
```

This means 880 non-DEGs and 120 DEGs satisfy FDR < 0.1. The plot function generates an M-A plot, where "M" indicates the log-ratio (i.e., $M = \log_2 G_2 - \log_2 G_1$) and "A" indicates average read count (i.e., $A = (\log_2 G_2 + \log_2 G_1)/2$), from the normalized count data. The magenta points indicate the identified DEGs at FDR < 0.1.
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., $AUC; 0 \leq AUC \leq 1$) 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).

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

[1] 0.8690469
```

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.

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

24
```r
> 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

<table>
<thead>
<tr>
<th>baseMean</th>
<th>log2FoldChange</th>
<th>lfcSE</th>
<th>stat</th>
<th>pvalue</th>
<th>padj</th>
</tr>
</thead>
<tbody>
<tr>
<td>gene_144</td>
<td>239.748</td>
<td>-1.97304</td>
<td>0.220284</td>
<td>-8.95678</td>
<td>3.34295e-19</td>
</tr>
<tr>
<td>gene_11</td>
<td>541.281</td>
<td>-1.94638</td>
<td>0.230809</td>
<td>-8.43287</td>
<td>3.73780e-17</td>
</tr>
<tr>
<td>gene_143</td>
<td>752.478</td>
<td>-1.816141</td>
<td>0.218271</td>
<td>-8.32181</td>
<td>8.66293e-17</td>
</tr>
<tr>
<td>gene_115</td>
<td>574.796</td>
<td>-1.78980</td>
<td>0.217648</td>
<td>-8.26749</td>
<td>8.67290e-16</td>
</tr>
<tr>
<td>gene_109</td>
<td>763.275</td>
<td>-1.72660</td>
<td>0.214623</td>
<td>-8.04432</td>
<td>8.67290e-16</td>
</tr>
</tbody>
</table>

> 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)

<table>
<thead>
<tr>
<th>gene_id</th>
<th>a.value</th>
<th>m.value</th>
<th>p.value</th>
<th>q.value</th>
<th>rank</th>
<th>estimatedDEG</th>
</tr>
</thead>
<tbody>
<tr>
<td>gene_168</td>
<td>8.905999</td>
<td>-1.811262</td>
<td>9.024627e-22</td>
<td>9.024627e-19</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>gene_144</td>
<td>7.591478</td>
<td>-1.97304</td>
<td>3.4295e-19</td>
<td>1.40571e-16</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>gene_11</td>
<td>8.774276</td>
<td>-1.94638</td>
<td>3.73780e-17</td>
<td>9.45070e-15</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>gene_143</td>
<td>8.905535</td>
<td>-1.78980</td>
<td>8.67290e-16</td>
<td>1.21565e-13</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

> 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
```
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.

```r
> 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.

```
log2 fold change (MLE): group 2 vs 1
Wald test p-value: group 2 vs 1
Data Frame with 6 rows and 6 columns

<table>
<thead>
<tr>
<th>gene</th>
<th>baseMean</th>
<th>log2FoldChange</th>
<th>lfcSE</th>
<th>stat</th>
<th>pvalue</th>
<th>padj</th>
</tr>
</thead>
<tbody>
<tr>
<td>gene_168</td>
<td>577.362</td>
<td>-1.81130</td>
<td>0.188923</td>
<td>-9.58750</td>
<td>9.02663e-22</td>
<td>7.58971e-19</td>
</tr>
<tr>
<td>gene_144</td>
<td>239.748</td>
<td>-1.97304</td>
<td>0.220284</td>
<td>-8.95678</td>
<td>3.34295e-19</td>
<td>1.40571e-16</td>
</tr>
<tr>
<td>gene_11</td>
<td>541.281</td>
<td>-1.94638</td>
<td>0.230809</td>
<td>-8.43287</td>
<td>3.37280e-17</td>
<td>9.45507e-15</td>
</tr>
<tr>
<td>gene_143</td>
<td>752.478</td>
<td>-1.81641</td>
<td>0.218271</td>
<td>-8.32181</td>
<td>3.37280e-17</td>
<td>1.82138e-14</td>
</tr>
<tr>
<td>gene_115</td>
<td>574.796</td>
<td>-1.78980</td>
<td>0.216423</td>
<td>-8.25749</td>
<td>8.66293e-16</td>
<td>2.50224e-14</td>
</tr>
<tr>
<td>gene_109</td>
<td>763.275</td>
<td>-1.72650</td>
<td>0.214623</td>
<td>-8.04432</td>
<td>8.67290e-16</td>
<td>1.21565e-13</td>
</tr>
</tbody>
</table>

> 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)

<table>
<thead>
<tr>
<th>gene_id</th>
<th>a.value</th>
<th>m.value</th>
<th>p.value</th>
<th>q.value</th>
<th>rank</th>
<th>estimatedDEG</th>
</tr>
</thead>
<tbody>
<tr>
<td>gene_151</td>
<td>9.737502</td>
<td>-2.742641</td>
<td>0.0001001188</td>
<td>0.01806983</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>gene_68</td>
<td>6.209168</td>
<td>-2.856852</td>
<td>0.0001338561</td>
<td>0.01806983</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>gene_11</td>
<td>8.772449</td>
<td>-2.094309</td>
<td>0.0001697872</td>
<td>0.01806983</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>gene_144</td>
<td>7.588239</td>
<td>-2.118569</td>
<td>0.0001859989</td>
<td>0.01806983</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>gene_16</td>
<td>7.348101</td>
<td>-2.331913</td>
<td>0.0001994984</td>
<td>0.01806983</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>gene_168</td>
<td>8.903234</td>
<td>-1.957337</td>
<td>0.0002068021</td>
<td>0.01806983</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

The DE results can be broken down as follows.

> table(tcc$estimatedDEG)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>879</td>
<td>121</td>
</tr>
</tbody>
</table>

This means 879 non-DEGs and 121 DEGs satisfy FDR < 0.1. The plot function generates an M-A plot, where "M" indicates the log-ratio (i.e., $M = \log_2 G_2 - \log_2 G_1$) and "A" indicates average read count (i.e., $A = (\log_2 G_2 + \log_2 G_1)/2$), 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; $0 \leq AUC \leq 1$) 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).

```r
> library(ROC)
> truth <- c(rep(1, 200), rep(0, 800))
> AUC(rocdemo.sca(truth = truth, data = -tcc$stat$rank))
[1] 0.8812469
```

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

```r
> 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

<table>
<thead>
<tr>
<th>gene</th>
<th>logFC</th>
<th>logCPM</th>
<th>F</th>
<th>PValue</th>
<th>FDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>gene_151</td>
<td>-2.572431</td>
<td>13.268838</td>
<td>100.18759</td>
<td>0.0001094357</td>
<td>0.02268321</td>
</tr>
<tr>
<td>gene_68</td>
<td>-2.756093</td>
<td>9.808735</td>
<td>91.34485</td>
<td>0.0001418618</td>
<td>0.02268321</td>
</tr>
<tr>
<td>gene_189</td>
<td>2.539469</td>
<td>9.661587</td>
<td>86.66409</td>
<td>0.0001608932</td>
<td>0.02268321</td>
</tr>
<tr>
<td>gene_196</td>
<td>2.501003</td>
<td>9.964259</td>
<td>85.83492</td>
<td>0.0001641463</td>
<td>0.02268321</td>
</tr>
<tr>
<td>gene_11</td>
<td>-1.953274</td>
<td>12.094808</td>
<td>77.78619</td>
<td>0.0002093873</td>
<td>0.02268321</td>
</tr>
<tr>
<td>gene_16</td>
<td>-2.290082</td>
<td>10.748393</td>
<td>75.60185</td>
<td>0.0002262868</td>
<td>0.02268321</td>
</tr>
</tbody>
</table>

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

```r
[1] 0.8637344
```

This is the same as

```r
> 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)
```

```r
gene_id  a.value  m.value  p.value  q.value  rank estimatedDEG
1 gene_151 9.740529 -2.600366 0.0001094357 0.02268321 1 1
2 gene_68  6.209636 -2.723799 0.0001418618 0.02268321 2 1
3 gene_189 6.127498  2.550316 0.0001608932 0.02268321 3 1
4 gene_196 6.461462  2.475704 0.0001641463 0.02268321 4 1
5 gene_11  8.774575 -1.957134 0.0002093873 0.02268321 5 1
6 gene_16  7.346399 -2.192904 0.0002262868 0.02268321 6 1
```

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

```r
[1] 0.8637344
```

### 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 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.

```r
> 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)

<table>
<thead>
<tr>
<th>gene_id</th>
<th>a.value</th>
<th>m.value</th>
<th>p.value</th>
<th>q.value</th>
<th>rank</th>
<th>estimatedDEG</th>
</tr>
</thead>
<tbody>
<tr>
<td>gene_121</td>
<td>NA</td>
<td>NA</td>
<td>2.851514e-07</td>
<td>0.0001158974</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>gene_64</td>
<td>NA</td>
<td>NA</td>
<td>3.681188e-07</td>
<td>0.0001158974</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>gene_140</td>
<td>NA</td>
<td>NA</td>
<td>5.936148e-07</td>
<td>0.0001158974</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>gene_88</td>
<td>NA</td>
<td>NA</td>
<td>6.528810e-07</td>
<td>0.0001158974</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>gene_134</td>
<td>NA</td>
<td>NA</td>
<td>7.348144e-07</td>
<td>0.0001158974</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>gene_74</td>
<td>NA</td>
<td>NA</td>
<td>9.230799e-07</td>
<td>0.0001158974</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>
```

> table(tcc$estimatedDEG)

```
0  1  
857 143
```

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 methods 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$count) with the simulateReadCounts function.
```r
> set.seed(1000)
> library(TCC)
> tcc <- simulateReadCounts(Ngene = 1000, PDEG = 0.2,
+ DEG.assign = c(0.9, 0.1),
+ DEG.foldchange = c(4, 4),
+ replicates = c(3, 3))
> dim(tcc$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     19     12     0
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 ($\mu$) and dispersion ($\phi$) 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 \texttt{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.

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

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>800</td>
<td>200</td>
</tr>
</tbody>
</table>

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.

```r
> 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  
gene_63  10.935474 -2.153519 2.275033e-06 0.001591835 1   1
gene_170 10.192416 -1.931026 4.373818e-06 0.001591835 2   1
gene_57  11.525601 -1.960298 1.034520e-05 0.001591835 3   1
gene_83  12.300687 -1.878646 1.090418e-05 0.001591835 4   1
gene_190 8.467250  1.973617 1.103774e-05 0.001591835 5   1
gene_23  8.701911 -2.121313 1.319053e-05 0.001591835 6   1
```

We can now calculate the area under the ROC curve (i.e., AUC; 0 ≤ AUC ≤ 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 \texttt{calcAUCValue} function calculates the AUC value based on the information stored in the \texttt{TCC} class object.

```r
> calcAUCValue(tcc)
```

```
[1] 0.9108312
```

This is essentially the same as

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

```
[1] 0.9108312
```

The following classic \texttt{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 \texttt{edgeR}.

33
> tcc <- calcNormFactors(tcc, norm.method = "tmm", iteration = 0)
> tcc <- estimateDE(tcc, test.method = "edgeR", FDR = 0.1)
> calcAUCValue(tcc)

[1] 0.89475

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] 0.89475

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

<table>
<thead>
<tr>
<th>group</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1_rep1</td>
</tr>
<tr>
<td>G1_rep2</td>
</tr>
<tr>
<td>G2_rep1</td>
</tr>
<tr>
<td>G2_rep2</td>
</tr>
<tr>
<td>G2_rep3</td>
</tr>
<tr>
<td>G2_rep4</td>
</tr>
<tr>
<td>G3_rep1</td>
</tr>
<tr>
<td>G3_rep2</td>
</tr>
<tr>
<td>G3_rep3</td>
</tr>
</tbody>
</table>

34
> head(tcc$count)

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

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

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

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

It can be seen that the DEGES/\texttt{edgeR}-\texttt{edgeR} combination outperforms the original \texttt{edgeR} procedure under the given simulation conditions. Note that the \texttt{test.method} argument will be ignored when \texttt{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., \texttt{Ngene = 10000}), (ii) the first 34\% of genes are DEGs (\texttt{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 (\texttt{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 (\texttt{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) (\texttt{replicates = c(1, 1, 2, 1, 1, 1, 1, 1)}).

```r
> 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))
```

> 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)

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

> tcc$group

```r
> tcc$group
```

```r
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)

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

```r
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
```

> head(tcc$count)

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

```r
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
```
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:

```r
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").

```r
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)`).

```r
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).

```r
head(tcc$count)
```

```r
    WT1d_rep1 WT1d_rep2 WT2d_rep1 WT2d_rep2 KO1d_rep1 KO1d_rep2 KO2d_rep1 KO2d_rep2
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     16    16
gene_5     2     0     1     2     1     4     0
```

```r
tcc$group
```

```r
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
```

38
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., \( N_{\text{gene}} = 20000 \)), (ii) the first 30% of genes are DEGs (\( P_{\text{DEG}} = 0.30 \)), (iii) the first 85% of the DEGs are up-regulated in G1 and the remaining 15% are up-regulated in G2 (\( \text{DEG.assign} = c(0.85, 0.15) \)), (iv) the levels of DE are eight-fold in G1 and sixteen-fold in G2 (\( \text{DEG.foldchange} = c(8, 16) \)), and (v) there are a total of four samples (two biological replicates for G1 and two biological replicates for G2) (\( \text{replicates} = c(2, 2) \)).

```
> 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
```
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)
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.

These upshifted M values for non-DEGs can be modified after performing the iDEGES/edgeR normalization, e.g., the median M value (= 0.066) for non-DEGs based on the iDEGES/edgeR-normalized data is nearly zero.
The comparison of those values obtained from different normalization methods might be another evaluation metric.
6  Session info

> sessionInfo()

R version 4.4.0 beta (2024-04-15 r86425)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 22.04.4 LTS

Matrix products: default
BLAS: /home/biocbuild/bbs-3.19-bioc/R/lib/libRblas.so
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0

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

time zone: America/New_York
tzcode source: system (glibc)

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

other attached packages:
[1] TCC_1.44.0 ROC_1.80.0
[3] edgeR_4.2.0 limma_3.60.0
[5] DESeq2_1.44.0 SummarizedExperiment_1.34.0
[7] Biobase_2.64.0 MatrixGenerics_1.16.0
[9] matrixStats_1.3.0 GenomicRanges_1.56.0
[11] GenomeInfoDb_1.40.0 IRanges_2.38.0
[13] S4Vectors_0.42.0 BiocGenerics_0.50.0

loaded via a namespace (and not attached):
[1] utf8_1.2.4         generics_0.1.3          SparseArray_1.4.0
[4] lattice_0.22-6    magrittr_2.0.3          grid_4.4.0
[7] jsonlite_1.8.8     Matrix_1.7-0            http_1.4.7
[10] fansi_1.0.6       UCSC.utils_1.0.0          scales_1.3.0
[13] codetools_0.2-20   abind_1.4-5             cli_3.6.2
[16] rlang_1.1.3       crayon_1.5.2            XVector_0.44.0
[19] splines_4.4.0     munsell_0.5.1            DelayedArray_0.30.0
[22] S4Arrays_1.4.0     tools_4.4.0             parallel_4.4.0
[25] BiocParallel_1.38.0 dplyr_1.1.4          colorspace_2.1-0
[28] ggplot2_3.5.1     locfit_1.5-9.9           GenomeInfoDbData_1.2.12
[31] vctrs_0.6.5        R6_2.5.1                lifecycle_1.0.4
[34] zlibbioc_1.50.0   pkgconfig_2.0.3          pillar_1.9.0
[37] glue_1.7.0         glue_1.7.0             Rcpp_1.0.12
[40] statsmod_1.5.0    xfun_0.43             tidyselect_1.2.1
[43] tidyselect_1.2.1   knitr_1.46             compiler_4.4.0
7 References


