

# Package ‘MultiRNAflow’

September 12, 2024

**Title** An R package for integrated analysis of temporal RNA-seq data with multiple biological conditions

**Version** 1.3.0

**Description** Our R package MultiRNAflow provides an easy to use unified framework allowing to automatically make both unsupervised and supervised (DE) analysis for datasets with an arbitrary number of biological conditions and time points. In particular, our code makes a deep downstream analysis of DE information, e.g. identifying temporal patterns across biological conditions and DE genes which are specific to a biological condition for each time.

**License** GPL-3 | file LICENSE

**URL** <https://github.com/loubator/MultiRNAflow>

**BugReports** <https://github.com/loubator/MultiRNAflow/issues>

**Depends** Mfuzz (>= 2.58.0), R (>= 4.3)

**Imports** Biobase (>= 2.54.0), ComplexHeatmap (>= 2.14.0), DESeq2 (>= 1.38.1), factoextra (>= 1.0.7), FactoMineR (>= 2.6), ggalluvial (>= 0.12.3), ggplot2 (>= 3.4.0), ggplotify (>= 0.1.2), ggrepel (>= 0.9.2), gprofiler2 (>= 0.2.1), graphics (>= 4.2.2), grDevices (>= 4.2.2), grid (>= 4.2.2), plot3D (>= 1.4), plot3Drgl (>= 1.0.3), reshape2 (>= 1.4.4), S4Vectors (>= 0.36.2), stats (>= 4.2.2), SummarizedExperiment (>= 1.28.0), UpSetR (>= 1.4.0), utils (>= 4.2.2)

**Suggests** BiocGenerics (>= 0.40.0), BiocStyle, e1071 (>= 1.7.12), knitr, rmarkdown, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**biocViews** Sequencing, RNASeq, GeneExpression, Transcription, TimeCourse, Preprocessing, Visualization, Normalization, PrincipalComponent, Clustering, DifferentialExpression, GeneSetEnrichment, Pathways

**Config/testthat/edition** 3

**Encoding** UTF-8

**LazyData** false

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

**git\_url** <https://git.bioconductor.org/packages/MultiRNAflow>

**git\_branch** devel

**git\_last\_commit** 850819a

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.20

**Date/Publication** 2024-09-11

**Author** Rodolphe Loubaton [aut, cre] (<<https://orcid.org/0000-0002-1442-7270>>),  
 Nicolas Champagnat [aut, ths] (<<https://orcid.org/0000-0002-5128-2357>>),  
 Laurent Vallat [aut, ths] (<<https://orcid.org/0000-0002-5226-7706>>),  
 Pierre Vallois [aut] (<<https://orcid.org/0000-0002-2123-0142>>),  
 Région Grand Est [fnd],  
 Cancéropôle Est [fnd]

**Maintainer** Rodolphe Loubaton <loubaton.rodolphe@gmail.com>

## Contents

MultiRNAflow-package . . . . .	3
CharacterNumbers . . . . .	4
ColnamesToFactors . . . . .	5
DATAnormalization . . . . .	6
DATAplotBoxplotSamples . . . . .	9
DATAplotExpression1Gene . . . . .	11
DATAplotExpressionGenes . . . . .	12
DATAprepSE . . . . .	14
DEanalysisGlobal . . . . .	17
DEanalysisGroup . . . . .	22
DEanalysisSubData . . . . .	25
DEanalysisTime . . . . .	26
DEanalysisTimeAndGroup . . . . .	29
DEplotAlluvial . . . . .	32
DEplotBarplot . . . . .	34
DEplotBarplotFacetGrid . . . . .	36
DEplotBarplotTime . . . . .	38
DEplotHeatmaps . . . . .	39
DEplotVennBarplotGroup . . . . .	41
DEplotVennBarplotTime . . . . .	43
DEplotVolcanoMA . . . . .	44
DEresultGroup . . . . .	46
DEresultGroupPerTime . . . . .	48
GSEAp preprocessing . . . . .	50
GSEAQuickAnalysis . . . . .	52
HCPCanalysis . . . . .	55
MFUZZanalysis . . . . .	58

MFUZZclustersNumber . . . . .	60
PCAanalysis . . . . .	63
PCAgraphics . . . . .	67
PCAp preprocessing . . . . .	70
PCArealization . . . . .	71
RawCountsSimulation . . . . .	73
RawCounts_Antoszewski2022_MOUSEsub500 . . . . .	74
RawCounts_Leong2014_FISSIONsub500wt . . . . .	75
RawCounts_Schleiss2021_CLLsub500 . . . . .	78
RawCounts_Weger2021_MOUSEsub500 . . . . .	82
Results_DEanalysis_sub500 . . . . .	87
Transcript_HomoSapiens_Database . . . . .	88

<b>Index</b>	<b>89</b>
--------------	-----------

---

MultiRNAflow-package    *MultiRNAflow: An R package for integrated analysis of temporal RNA-seq data with multiple biological conditions*

---

## Description

Our R package MultiRNAflow provides an easy to use unified framework allowing to automatically make both unsupervised and supervised (DE) analysis for datasets with an arbitrary number of biological conditions and time points. In particular, our code makes a deep downstream analysis of DE information, e.g. identifying temporal patterns across biological conditions and DE genes which are specific to a biological condition for each time.

## Details

The main functions are:

- [DATAnormalization](#) - to normalize raw count data
- [PCAanalysis](#) - to perform PCA analysis with `FactoMineR::PCA()`
- [HCPCanalysis](#) - to perform hierarchical clustering with `FactoMineR::HCPC()`
- [MFUZZanalysis](#) - to perform temporal clustering with `Mfuzz::mfuzz.plot2()`
- [DEanalysisGlobal](#) - to perform differential analysis with `DESeq2::DESeq()`
- [GSEAQuickAnalysis](#) - to perform enrichment analysis with `gprofiler2::gost()`
- [GSEAp preprocessing](#) - to return preprocessing file for official software and online tools performing enrichment analysis

## Author(s)

**Maintainer:** Rodolphe Loubaton <loubaton.rodolphe@gmail.com> ([ORCID](#))

Authors:

- Nicolas Champagnat <nicolas.champagnat@inria.fr> ([ORCID](#)) [thesis advisor]

- Laurent Vallat <vallat@unistra.fr> ([ORCID](#)) [thesis advisor]
- Pierre Vallois <pierre.vallois@univ-lorraine.fr> ([ORCID](#))

Other contributors:

- Région Grand Est [funder]
- Cancéropôle Est [funder]

## See Also

Useful links:

- <https://github.com/loubator/MultiRNAflow>
- Report bugs at <https://github.com/loubator/MultiRNAflow/issues>

---

CharacterNumbers	<i>Transformation of a vector of integers into a vector of class "character".</i>
------------------	---

---

## Description

Transformation of a vector of integers into a vector of class "character" so that lexicographic order of characters corresponds to the numerical order of time measurements.

## Usage

```
CharacterNumbers(Vect.number)
```

## Arguments

Vect.number      Vector of integers.

## Details

An appropriate number of character "0" is added in front of the numerical characters corresponding to the decimal writing of each integer in Vect.number so that the order of elements of the vector is preserved. For example, "9">"11", but "09"<"11".

## Value

A vector where each integer is transformed in class "character".

## See Also

The function is called by [ColnamesToFactors\(\)](#).

## Examples

```
CharacterNumbers(Vect.number=c(0,1,9,11,90,99,100,101))
CharacterNumbers(Vect.number=0:11)
CharacterNumbers(Vect.number=1:8)
```

---

ColnamesToFactors	<i>Extraction of factors information and suitable column names creation from the column names of a dataset.</i>
-------------------	---

---

## Description

This function generates new reduced column names according to the presence of biological conditions and/or time points, and extract the different factors (individual's names, time measurements, biological conditions) from the column names of the dataset (see Details).

## Usage

```
ColnamesToFactors(
  ExprData,
  Column.gene,
  Group.position,
  Time.position,
  Individual.position
)
```

## Arguments

ExprData	Data.frame with $N_g$ rows and $(N_{s+k})$ columns, where $N_g$ is the number of genes, $N_s$ is the number of samples and $k = 1$ if a column is used to specify gene names, or $k = 0$ otherwise. If $k = 1$ , the position of the column containing gene names is given by Column.gene. The data.frame contains numeric values giving gene expressions of each gene in each sample. Gene expressions can be raw counts or normalized raw counts. Column names of the data.frame must describe each sample's information (individual, biological condition and time) and have the structure described in the section Details.
Column.gene	Integer indicating the column where gene names are given. Set Column.gene=NULL if there is no such column.
Group.position	Integer indicating the position of group information in the string of characters in each sample names (see Details). Set Group.position=NULL if there is only one or no biological information in the string of character in each sample name.
Time.position	Integer indicating the position of time measurement information in the string of characters in each sample names (see Details). Set Time.position=NULL if there is only one or no time measurement information in the string of character in each sample name.
Individual.position	Integer indicating the position of the name of the individual (e.g patient, replicate, mouse, yeasts culture ...) in the string of characters in each sample names (see Details). The names of different individuals must be all different. Furthermore, if individual names are just numbers, they will be transform in a vector of class "character" by <a href="#">CharacterNumbers()</a> and a "r" will be added to each individual name ("r" for replicate).

## Details

The column names of `ExprData` must be a vector of strings of characters containing

- a string of characters (if  $k = 1$ ) which is the label of the column containing gene names.
- $N_s$  sample names which must be strings of characters containing at least : the name of the individual (e.g patient, mouse, yeasts culture), its biological condition (if there is at least two) and the time where data have been collected if there is at least two; (must be either 't0', 'T0' or '0' for time 0, 't1', 'T1' or '1' for time 1, ...).

All these sample information must be separated by underscores in the sample name. For instance 'CLL\_P\_t0\_r1', corresponds to the patient 'r1' belonging to the biological condition 'P' and where data were collected at time 't0'. In this example, 'CLL' describe the type of cells (here chronic lymphocytic leukemia) and is not used in our analysis.

In the string of characters 'CLL\_P\_t0\_r1', 'r1' is localized after the third underscore, so `Individual.position=4`, 'P' is localized after the first underscore, so `Group.position=2` and 't0' is localized after the second underscore, so `Time.position=3`.

## Value

The function returns new column names of the dataset, a vector indicating the name of the individual for each sample, a vector indicating the time for each sample and/or a vector indicating the biological condition for each sample.

## See Also

The `ColnamesToFactors()` function is used by the following functions of our package : `DATAprepSE()`, `PCAprereprocessing()`, `MFUZZclustersNumber()` and `MFUZZanalysis()`.

## Examples

```
## Data simulated with our function RawCountsSimulation()
Data.sim <- RawCountsSimulation(Nb.Group=3, Nb.Time=2, Nb.per.GT=3,
                               Nb.Gene=10)
##-----##
res.test.colnames <- ColnamesToFactors(ExprData=Data.sim$Sim.dat,
                                       Column.gene=1,
                                       Group.position=1,
                                       Time.position=2,
                                       Individual.position=3)

print(res.test.colnames)
```

## Description

From raw counts, this function realizes one of the three methods of normalization of the package DESeq2:

- Relative Log Expression (rle) transformation (see `BiocGenerics::estimateSizeFactors()`)
- Regularized Log (rlog) transformation (see `DESeq2::rlog()`)
- Variance Stabilizing Transformation (vst) transformation (see `DESeq2::vst()`)

## Usage

```
DATAnormalization(
  SEres,
  Normalization = "vst",
  Blind.rlog.vst = FALSE,
  Plot.Boxplot = TRUE,
  Colored.By.Factors = FALSE,
  Color.Group = NULL,
  Plot.genes = FALSE,
  path.result = NULL,
  Name.folder.norm = NULL
)
```

## Arguments

SEres	Results of the function <code>DATAprepSE()</code> .
Normalization	"rle", "vst", "rlog" and "rpkm". "rle", "vst" and "rlog" correspond to a method of normalization proposed by DESeq2 (see <code>BiocGenerics::estimateSizeFactors()</code> for "rle", <code>DESeq2::rlog()</code> for "rlog" and <code>DESeq2::vst()</code> for "vst"). "rpkm" corresponds to a RPKM normalization after a "rle" normalization.
Blind.rlog.vst	TRUE or FALSE. FALSE by default. See input 'blind' in <code>DESeq2::rlog()</code> . It is recommended to set <code>Blind.rlog.vst=FALSE</code> for downstream analysis.
Plot.Boxplot	TRUE or FALSE. TRUE by default. If <code>Plot.Boxplot=TRUE</code> , the function <code>DATApplotBoxplotSamples()</code> will be called and boxplots will be plotted. Otherwise, no boxplots will be plotted.
Colored.By.Factors	TRUE or FALSE. FALSE by default. If TRUE, boxplots will be colored with different colors for different time measurements (if data were collected at different time points). Otherwise, boxplots will be colored with different colors for different biological conditions.
Color.Group	NULL or a data.frame with $N_{bc}$ rows and two columns where $N_{bc}$ is the number of biological conditions. NULL by default. If <code>Color.Group</code> is a data.frame, the first column must contain the name of each biological condition and the second column must contain the colors associated to each biological condition. If <code>Color.Group=NULL</code> , the function will automatically attribute a color for each biological condition. If samples belong to different time points only, <code>Color.Group</code> will not be used.

<code>Plot.genes</code>	TRUE or FALSE. FALSE by default. If TRUE, points representing gene expressions (normalized or raw counts) will be plotted for each sample. Otherwise, only boxplots will be plotted.
<code>path.result</code>	Character or NULL. NULL by default. Path to save all results. If <code>path.result</code> contains a sub folder entitled "1_Normalization_Name.folder.norm" all results will be saved in the sub folder "1_Normalization_Name.folder.norm". Otherwise, a sub folder entitled "1_Normalization_Name.folder.norm" will be created in <code>path.result</code> and all results will be saved in "1_Normalization_Name.folder.norm". If NULL, the results will not be saved in a folder. NULL as default.
<code>Name.folder.norm</code>	Character or NULL. NULL by default. If <code>Name.folder.norm</code> is a character, the folder name which will contain the results will be "1_Normalization_Name.folder.norm". Otherwise, the folder name will be "1_Normalization".

### Details

All results are built from the results of the function [DATAprepSE\(\)](#).

### Value

The function returns a SummarizedExperiment object (SEresNorm) identical as SEres but

- with the normalized count data saved in `assays(SEresNORM)[[2]]`
- with the boxplot of normalized count saved in the metadata `Results[[1]][[1]]` of SEresNorm.

The boxplot is plotted if `Plot.Boxplot=TRUE`.

### See Also

The [DATAnormalization\(\)](#) function calls our R function [DATAprepSE\(\)](#), and the R functions [BiocGenerics::estimateSizeFactors\(\)](#), [DESeq2::rlog\(\)](#) and [DESeq2::vst\(\)](#) in order to re-alized the normalization.

### Examples

```
data(RawCounts_Antoszewski2022_MOUSEsub500)
##-----##
resDATAprepSE <- DATAprepSE(RawCounts=RawCounts_Antoszewski2022_MOUSEsub500,
                             Column.gene=1,
                             Group.position=1,
                             Time.position=NULL,
                             Individual.position=2)
##-----##
resNorm <- DATAnormalization(SEres=resDATAprepSE,
                             Normalization="rle",
                             Plot.Boxplot=TRUE,
                             Colored.By.Factors=TRUE)
```



---

 DATAplotBoxplotSamples

*Visualization of the distribution of all gene expressions using a boxplot for each sample.*

---

## Description

From the results of either our R function `DATAprepSE()` or our R function `DATAnormalization()` (raw counts or normalized raw counts), the function plots the distribution of all gene expressions using a boxplot for each sample.

## Usage

```
DATAplotBoxplotSamples(
  SEres,
  Log2.transformation = TRUE,
  Colored.By.Factors = FALSE,
  Color.Group = NULL,
  Plot.genes = FALSE,
  y.label = NULL
)
```

## Arguments

<code>SEres</code>	Results of the function <code>DATAprepSE()</code> or <code>DATAnormalization()</code> .
<code>Log2.transformation</code>	TRUE or FALSE. TRUE by default. If TRUE, each numeric value $x$ in <code>ExprData</code> will become $\log_2(x + 1)$ (see Details).
<code>Colored.By.Factors</code>	TRUE or FALSE. FALSE by default. If TRUE, boxplots will be colored with different colors for different time measurements (if data were collected at different time points). Otherwise, boxplots will be colored with different colors for different biological conditions.
<code>Color.Group</code>	NULL or a data.frame with $N_{bc}$ rows and two columns where $N_{bc}$ is the number of biological conditions. NULL by default. If <code>Color.Group</code> is a data.frame, the first column must contain the name of each biological condition and the second column must contain the colors associated to each biological condition. If <code>Color.Group=NULL</code> , the function will automatically attribute a color for each biological condition. If samples belong to different time points only, <code>Color.Group</code> will not be used.
<code>Plot.genes</code>	TRUE or FALSE. FALSE by default. If TRUE, points representing gene expression (normalized or raw counts) will be added for each sample.
<code>y.label</code>	NULL or a character. NULL by default. If <code>y.label</code> is a character, it will be the y label of the graph. If <code>y.label=NULL</code> , the label will be either " $\log_2(\text{Gene expression} + 1)$ " (if <code>Log2.transformation=TRUE</code> ) or "Gene expression" (if <code>Log2.transformation=FALSE</code> ).

## Details

The boxplot allows to visualize six summary statistics (see [ggplot2::geom\\_boxplot\(\)](#)):

- the median
- two hinges: first and third quartiles denoted Q1 and Q3.
- two whiskers:  $W1 := Q1 - 1.5 * IQR$  and  $W3 := Q3 + 1.5 * IQR$  with  $IQR = Q3 - Q1$ , the interquartile range.
- outliers: data beyond the end of the whiskers are called "outlying" points and are plotted in black.

For better visualization of the six summary statistics described above, raw counts must be transformed using the function  $\log_2(x + 1)$ . This transformation is automatically performed by other functions of the package, such as [DATAnormalization\(\)](#). Log2.transformation will be set as TRUE in [DATAnormalization\(\)](#) if Normalization = "rle", otherwise Log2.transformation=FALSE.

## Value

The function returns a graph which plots the distribution of all gene expressions using a boxplot for each sample (see [ggplot2::geom\\_boxplot\(\)](#)).

## See Also

The [DATAplotBoxplotSamples\(\)](#) function

- is used by the following function of our package: [DATAnormalization\(\)](#).
- calls the R functions [ggplot2::geom\\_boxplot](#) and [ggplot2::geom\\_jitter](#) in order to print the boxplot.

## Examples

```
data(RawCounts_Antoszewski2022_MOUSEsub500)
##-----##
resDATAprepSE <- DATAprepSE(RawCounts=RawCounts_Antoszewski2022_MOUSEsub500,
                             Column.gene=1,
                             Group.position=1,
                             Time.position=NULL,
                             Individual.position=2)
##-----##
DATAplotBoxplotSamples(SEres=resDATAprepSE,
                       Log2.transformation=TRUE,
                       Colored.By.Factors=TRUE,
                       Color.Group=NULL,
                       Plot.genes=FALSE,
                       y.label=NULL)
```

---

 DATAplotExpression1Gene

*Plot expression of one gene.*


---

### Description

The function allows to plot the gene expression profile of one gene only according to time and/or biological conditions.

### Usage

```
DATAplotExpression1Gene(
  SEres,
  row.gene = 1,
  Color.Group = NULL,
  ylabel = "Expression"
)
```

### Arguments

SEres	Results of either our R function <a href="#">DATAprepSE()</a> , or our R function <a href="#">DATAnormalization()</a> .
row.gene	Non negative integer indicating the row of the gene to be plotted.
Color.Group	NULL or a data.frame with $N_{bc}$ rows and two columns where $N_{bc}$ is the number of biological conditions. If Color.Group is a data.frame, the first column must contain the name of each biological condition and the second column must contain the colors associated to each biological condition. If Color.Group=NULL, the function will automatically attribute a color for each biological condition. If samples belong to different time points only, Color.Group will not be used.
ylabel	Character corresponding to the label of the y-axis. By default, ylab="Expression".

### Details

All results are built from either the results of our R function [DATAprepSE\(\)](#) or the results of our R function [DATAnormalization\(\)](#).

### Value

The function plots for the gene selected with the input row.gene

- In the case where samples belong to different time points only : the evolution of the expression of each replicate across time and the evolution of the mean and the standard deviation of the expression across time.
- In the case where samples belong to different biological conditions only: a violin plot (see [ggplot2::geom\\_violin\(\)](#)), and error bars (standard deviation) (see [ggplot2::geom\\_errorbar\(\)](#)) for each biological condition.
- In the case where samples belong to different time points and different biological conditions : the evolution of the expression of each replicate across time and the evolution of the mean and the standard deviation of the expression across time for each biological condition.

**See Also**

The `DATAplotExpression1Gene()` function is used by the following function of our package: `DATAplotExpressionGenes()`.

**Examples**

```
## Simulation raw counts
resSIMcount <- RawCountsSimulation(Nb.Group=2, Nb.Time=3, Nb.per.GT=4,
                                   Nb.Gene=10)

## Preprocessing step
resDATAprepSE <- DATAprepSE(RawCounts=resSIMcount$Sim.dat,
                             Column.gene=1,
                             Group.position=1,
                             Time.position=2,
                             Individual.position=3)

##-----##
resEV01gene <- DATAplotExpression1Gene(SERes=resDATAprepSE,
                                       row.gene=1,
                                       Color.Group=NULL)

print(resEV01gene)
```

---

DATAplotExpressionGenes

*Plot expression of a subset of genes.*

---

**Description**

The function allows to plot gene expression profiles according to time and/or biological conditions.

**Usage**

```
DATAplotExpressionGenes(
  SEResNorm,
  Vector.row.gene,
  DATAnorm = TRUE,
  Color.Group = NULL,
  Plot.Expression = TRUE,
  path.result = NULL,
  Name.folder.profile = NULL
)
```

**Arguments**

`SEResNorm` Results of the function `DATANormalization()`.  
`Vector.row.gene` Vector of non negative integers indicating the rows of the genes to be plotted.

<code>DATAnorm</code>	TRUE or FALSE. TRUE by default. TRUE means the function plots gene normalized expression profiles. FALSE means the function plots gene raw expression profiles.
<code>Color.Group</code>	NULL or a data.frame with $N_{bc}$ rows and two columns where $N_{bc}$ is the number of biological conditions. If <code>Color.Group</code> is a data.frame, the first column must contain the name of each biological condition and the second column must contain the colors associated to each biological condition. If <code>Color.Group=NULL</code> , the function will automatically attribute a color for each biological condition. If samples belong to different time points only, <code>Color.Group</code> will not be used. NULL by default.
<code>Plot.Expression</code>	TRUE or FALSE. TRUE by default. If TRUE, the graph will be plotted. Otherwise no graph will be plotted.
<code>path.result</code>	Character or NULL. NULL by default. Path to save all results. If <code>path.result</code> contains a sub folder entitled " <code>1_UnsupervisedAnalysis_Name.folder.profile</code> " and a sub sub folder, " <code>1-5_ProfileExpression_Name.folder.profile</code> " all results will be saved in the sub folder " <code>1_UnsupervisedAnalysis_Name.folder.profile/1-5_ProfileExpression_Name.folder.profile</code> ". Otherwise, a sub folder entitled " <code>1_UnsupervisedAnalysis_Name.folder.profile</code> " and/or a sub sub folder " <code>1-5_ProfileExpression_Name.folder.profile</code> " will be created in <code>path.result</code> and all results will be saved in " <code>1_UnsupervisedAnalysis_Name.folder.profile/1-5_ProfileExpression_Name.folder.profile</code> ". If NULL, the results will not be saved in a folder. NULL as default.
<code>Name.folder.profile</code>	Character or NULL. NULL by default. If <code>Name.folder.profile</code> is a character, the folder and sub folder names which will contain the PCA graphs will respectively be " <code>1_UnsupervisedAnalysis_Name.folder.profile</code> " and " <code>1-5_ProfileExpression_Name.folder.profile</code> ". Otherwise, the folder and sub folder names will respectively be " <code>1_UnsupervisedAnalysis</code> " and " <code>1-5_ProfileExpression</code> ".

## Details

All results are built from the results of our function `DATAnormalization()`.

## Value

The function returns the same `SummarizedExperiment` class object `SEresNorm` with the a graph for each gene (depending on the experimental design) selected with the input `Vector.row.gene` (saved in the metadata `Results[[1]][[5]]` of `SEresNorm`)

- In the case where samples belong to different time points only : the evolution of the expression of each replicate across time and the evolution of the mean and the standard deviation of the expression across time.
- In the case where samples belong to different biological conditions only: a violin plot (see `ggplot2::geom_violin()`), and error bars (standard deviation) (see `ggplot2::geom_errorbar()`) for each biological condition.
- In the case where samples belong to different time points and different biological conditions : the evolution of the expression of each replicate across time and the evolution of the mean and the standard deviation of the expression across time for each biological condition.

The function plots the different graph if `Plot.Expression=TRUE`.

### See Also

The function calls our R function [DATAnormalization\(\)](#) first, then [DATAplotExpression1Gene\(\)](#) for each selected genes with `Vector.row.gene`.

### Examples

```
## Simulation raw counts
resSIMcount <- RawCountsSimulation(Nb.Group=2, Nb.Time=3, Nb.per.GT=4,
                                   Nb.Gene=10)

## Preprocessing step
resDATAprepSE <- DATAprepSE(RawCounts=resSIMcount$Sim.dat,
                             Column.gene=1,
                             Group.position=1,
                             Time.position=2,
                             Individual.position=3)

## Normalization
resNorm <- DATAnormalization(SEres=resDATAprepSE,
                              Normalization="rle",
                              Plot.Boxplot=FALSE,
                              Colored.By.Factors=FALSE)

##-----##
resEVOgenes <- DATAplotExpressionGenes(SEresNorm=resNorm,
                                       Vector.row.gene=c(1,3),
                                       DATAnorm=TRUE,
                                       Color.Group=NULL,
                                       Plot.Expression=TRUE,
                                       path.result=NULL,
                                       Name.folder.profile=NULL)
```

---

DATAprepSE

*Data preparation for exploratory and statistical analysis (Main Function)*

---

### Description

This function creates automatically a `SummarizedExperiment` (SE) object from raw counts data to store

- information for exploratory (unsupervised) analysis using the R function [SummarizedExperiment::SummarizedExperimentFromMatrix\(\)](#)
- a `DESeq2` object from raw counts data in order to store all information for statistical (supervised) analysis using the R function [DESeq2::DESeqDataSetFromMatrix\(\)](#).

**Usage**

```

DATAprepSE(
  RawCounts,
  Column.gene,
  Group.position,
  Time.position,
  Individual.position,
  colData = NULL,
  VARfilter = 0,
  SUMfilter = 0,
  RNAlength = NULL
)

```

**Arguments**

- |                     |   |
|---------------------|---|
| RawCounts           | Data.frame with $N_g$ rows and $(N_s + k)$ columns, where $N_g$ is the number of genes, $N_s$ is the number of samples and $k = 1$ if a column is used to specify gene names, or $k = 0$ otherwise. If $k = 1$ , the position of the column containing gene names is given by Column.gene. The data.frame contains non negative integers giving gene expressions of each gene in each sample. Column names of the data.frame must describe each sample's information (individual, biological condition and time) and have the structure described in the section Details.   |
| Column.gene         | Integer indicating the column where gene names are given. Set Column.gene=NULL if there is no such column.  |
| Group.position      | Integer indicating the position of group information in the string of characters in each sample names (see Details). Set Group.position=NULL if there is only one or no biological information in the string of character in each sample name.  |
| Time.position       | Integer indicating the position of time measurement information in the string of characters in each sample names (see Details). Set Time.position=NULL if there is only one or no time measurement information in the string of character in each sample name.  |
| Individual.position | Integer indicating the position of the name of the individual (e.g patient, replicate, mouse, yeasts culture ...) in the string of characters in each sample names (see Details). The names of different individuals must be all different. Furthermore, if individual names are just numbers, they will be transform in a vector of class "character" by <a href="#">CharacterNumbers()</a> and a "r" will be added to each individual name ("r" for replicate).   |
| colData             | <p>NULL or data.frame with <math>N_s</math> rows and two or three columns describing the samples. NULL as default. Optional input (see Details). If Group.position, Time.position and Individual.position are filled, set colData=NULL.</p> <ul style="list-style-type: none"> <li>• If samples belong to different times point and different biological condition           <ul style="list-style-type: none"> <li>– the first column must contain the biological condition for each sample. The column name must be "Group".</li> <li>– the second column must contain the time measurement for each sample. The column name must be "Time".</li> </ul> </li> </ul> |

	<ul style="list-style-type: none"> <li>– The third column must contain the individual name for each sample. The column name must be "ID".</li> <li>• If samples belong to different times point or different biological condition <ul style="list-style-type: none"> <li>– the first column must contain, either the biological condition for each sample, or the time measurement for each sample. The column name must be either "Group", or "Time".</li> <li>– The second column must contain the individual name for each sample. The column name must be "ID".</li> </ul> </li> </ul>
VARfilter	Positive numeric value, 0 as default. All rows of RawCounts which the variance of counts is strictly under the threshold VARfilter are deleted
SUMfilter	Positive numeric value, 0 as default. All rows of RawCounts which the sum of counts is strictly under the threshold SUMfilter are deleted.
RNAlength	<p>NULL or "hsapiens" or data.frame with two columns. NULL as default.</p> <ul style="list-style-type: none"> <li>• if RNAlength is a data.frame <ul style="list-style-type: none"> <li>– the first column must contain gene names (similar to those of RawCounts)</li> <li>– the second columns must contain the median of the transcript length of each gene of the first column and all rows of RawCounts whose genes are not included in the first column of RNAlength will be deleted.</li> </ul> </li> <li>• if RNAlength=NULL, no rows will be deleted.</li> </ul> <p>If RNAlength is either "hsapiens" or a data.frame, Column.gene can not be NULL.</p>

## Details

The column names of RawCounts must be a vector of strings of characters containing

- a string of characters (if  $k = 1$ ) which is the label of the column containing gene names.
- $N_s$  sample names which must be strings of characters containing at least : the name of the individual (e.g patient, mouse, yeasts culture), its biological condition (if there is at least two) and the time where data have been collected if there is at least two; (must be either 't0', 'T0' or '0' for time 0, 't1', 'T1' or '1' for time 1, ...).

All these sample information must be separated by underscores in the sample name. For instance 'CLL\_P\_t0\_r1', corresponds to the patient 'r1' belonging to the biological condition 'P' and where data were collected at time 't0'. In this example, 'CLL' describe the type of cells (here chronic lymphocytic leukemia) and is not used in our analysis.

In the string of characters 'CLL\_P\_t0\_r1', 'r1' is localized after the third underscore, so Individual.position=4, 'P' is localized after the first underscore, so Group.position=2 and 't0' is localized after the second underscore, so Time.position=3.

If the user does not have all these sample information separated by underscores in the sample name, the user can build the data.frame colData describing the samples.

## Value

The function returns a SummarizedExperiment object containing all information for exploratory (unsupervised) analysis and DE statistical analysis.



**See Also**

The `DATAprepSE()` function

- is used by the following functions of our package : `DATAnormalization()`, `DEanalysisGlobal()`.
- calls the R function `DESeq2::DESeqDataSetFromMatrix()` in order to create the `DESeq2` object and `SummarizedExperiment::SummarizedExperiment()` in order to create the `SummarizedExperiment` object

**Examples**

```
BgCdEx <- rep(c("P", "NP"), each=27)
TimeEx <- rep(paste0("t", seq_len(9) - 1), times=6)
IndvEx <- rep(paste0("pc1", seq_len(6)), each=9)

SampleNAMEEx <- paste(BgCdEx, IndvEx, TimeEx, sep="_")
RawCountEx <- data.frame(Gene.name=paste0("Name", seq_len(10)),
                        matrix(sample(seq_len(100),
                                    length(SampleNAMEEx)*10,
                                    replace=TRUE),
                                ncol=length(SampleNAMEEx), nrow=10))
colnames(RawCountEx) <- c("Gene.name", SampleNAMEEx)
##-----##
resDATAprepSE <- DATAprepSE(RawCounts=RawCountEx,
                          Column.gene=1,
                          Group.position=1,
                          Time.position=3,
                          Individual.position=2)

##
## colDataEx <- data.frame(Group=BgCdEx, Time=TimeEx, ID=IndvEx)
```

---

DEanalysisGlobal      *Realization of the DE analysis (Main Function).*

---

**Description**

The function realizes the DE analysis in three cases: either samples belonging to different time measurements, or samples belonging to different biological conditions, or samples belonging to different time measurements and different biological conditions.

**Usage**

```
DEanalysisGlobal(
  SRes,
  pval.min = 0.05,
  pval.vect.t = NULL,
  log.FC.min = 1,
  LRT.supp.info = FALSE,
  Plot.DE.graph = TRUE,
```

```

    path.result = NULL,
    Name.folder.DE = NULL
  )

```

### Arguments

<code>SEres</code>	Results of either our R function <a href="#">DATAprepSE()</a> , or our R function <a href="#">DATAnormalization()</a> .
<code>pval.min</code>	Numeric value between 0 and 1. A gene will be considered as differentially expressed (DE) between two biological conditions if its Benjamini-Hochberg adjusted p-value (see <a href="#">stats::p.adjust()</a> ) is below the threshold <code>pval.min</code> . Default value is 0.05.
<code>pval.vect.t</code>	NULL or vector of dimension $T - 1$ filled with numeric values between 0 and 1, with $T$ the number of time measurements. A gene will be considered as differentially expressed (DE) between the time $t_i$ and the reference time $t_0$ if its Benjamini-Hochberg adjusted p-value (see <a href="#">stats::p.adjust()</a> ) is below the $i$ -th threshold of <code>pval.vect.t</code> . If NULL, <code>pval.vect.t</code> will be vector of dimension $T - 1$ filled with <code>pval.min</code> .
<code>log.FC.min</code>	Non negative numeric value. If the $\log_2$ fold change between biological conditions or times has an absolute value below the threshold <code>log.FC.min</code> , then the gene is not selected even if is considered as DE. Default value is 1. If <code>log.FC.min=0</code> , all DE genes will be kept.
<code>LRT.suppl.info</code>	TRUE or FALSE. If TRUE, the algorithm realizes another statistical test in order to detect if, among all biological conditions and/or times, at least one has a different behavior than the others (see the input test in <a href="#">DESeq2::DESeq()</a> ).
<code>Plot.DE.graph</code>	TRUE or FALSE. TRUE as default. If TRUE, all graphs will be plotted. Otherwise no graph will be plotted.
<code>path.result</code>	Character or NULL. Path to save all results. If <code>path.result</code> contains a sub folder entitled "DEanalysis_Name.folder.DE" all results will be saved in the sub folder "DEanalysis_Name.folder.DE". Otherwise, a sub folder entitled "DEanalysis_Name.folder.DE" will be created in <code>path.result</code> and all results will be saved in "DEanalysis_Name.folder.DE". If NULL, the results will not be saved in a folder. NULL as default.
<code>Name.folder.DE</code>	Character or NULL. If <code>Name.folder.DE</code> is a character, the folder names which will contain all results will be "DEanalysis_Name.folder.DE". Otherwise, the folder name will be "DEanalysis".

### Details

All results are built from the results of either our R function [DATAprepSE\(\)](#), or our R function [DATAnormalization\(\)](#).

### Value

The function returns the same SummarizedExperiment class object `SEres` with the rle normalized count data (cf [DATAnormalization\(\)](#)) automatically realized by [DESeq2::DESeq\(\)](#) and saved in `assays(SEresNORM)$rle`, and with the following results saved in the metadata `Results[[2]][[2]]` of `SEres`, depending on the experimental design.

- If samples belong to different biological conditions only (see [DEanalysisGroup\(\)](#)), the function returns
  - a data.frame (output `rowData(Seres)`) which contains
    - \* pvalues, log2 fold change and DE genes between each pairs of biological conditions.
    - \* a binary column (1 and 0) where 1 means the gene is DE between at least one pair of biological conditions.
    - \*  $N_{bc}$  binary columns, where  $N_{bc}$  is the number of biological conditions, which gives the specific genes for each biological condition. A '1' in one of these columns means the gene is specific to the biological condition associated to the given column. 0 otherwise. A gene is called specific to a given biological condition BC1, if the gene is DE between BC1 and any other biological conditions, but not DE between any pair of other biological conditions.
    - \*  $N_{bc}$  columns filled with -1, 0 and 1, one per biological condition. A '1' in one of these columns means the gene is up-regulated (or over-expressed) for the biological condition associated to the given column. A gene is called up-regulated for a given biological condition BC1 if the gene is specific to the biological condition BC1 and expressions in BC1 are higher than in the other biological conditions. A '-1' in one of these columns means the gene is down-regulated (or under-expressed) for the biological condition associated to the given column. A gene is called down-regulated for a given biological condition BC1 if the gene is specific to the biological condition BC1 and expressions in BC1 are lower than in the other biological conditions. A '0' in one of these columns means the gene is not specific to the biological condition associated to the given column.
  - an UpSet plot (Venn diagram displayed as a barplot) which gives the number of genes for each possible intersection (see [DEplotVennBarplotGroup\(\)](#)). We consider that a set of pairs of biological conditions forms an intersection if there is at least one gene which is DE for each of these pairs of biological conditions, but not for the others.
  - a barplot which gives the number of genes categorized as "Upregulated" and "Down-Rugulated", per biological condition (see [DEplotBarplot\(\)](#)).
  - a barplot which gives the number of genes categorized as "Upregulated", "DownRugulated" and "Other", per biological condition (see [DEplotBarplot\(\)](#)). A gene is categorized as 'Other', for a given biological condition, if the gene is not specific to the given biological condition. So this barplot, only plotted when there are strictly more than two biological conditions, is similar to the previous barplot but with the category "Other".
  - a list (output `List.Glossary`) containing the glossary of the column names of `DE.results`.
  - a list (output `Summary.Inputs`) containing a summary of sample information and inputs of [DEanalysisGlobal\(\)](#).
- If data belong to different time points only (see [DEanalysisTime\(\)](#)), the function returns
  - a data.frame (output `rowData(Seres)`) which contains
    - \* gene names
    - \* pvalues, log2 fold change and DE genes between each time  $t_i$  versus the reference time  $t_0$ .
    - \* a binary column (1 and 0) where 1 means the gene is DE at at least between one time  $t_i$  versus the reference time  $t_0$ .
    - \* a column where each element is succession of 0 and 1. The positions of '1' indicate the set of times  $t_i$  such that the gene is DE between  $t_i$  and the reference time  $t_0$ .

- an alluvial graph of differentially expressed (DE) genes (see `DEplotAlluvial()`)
  - a graph showing the number of DE genes as a function of time for each temporal group (see `DEplotAlluvial()`). By temporal group, we mean the sets of genes which are first DE at the same time.
  - a barplot which gives the number of DE genes per time (see `DEplotBarplotTime()`)
  - an UpSet plot which gives the number of genes per temporal pattern (see `DEplotVennBarplotTime()`). By temporal pattern, we mean the set of times  $t_i$  such that the gene is DE between  $t_i$  and the reference time  $t_0$ .
  - a similar UpSet plot where each bar is split in different colors corresponding to all possible numbers of DE times where genes are over expressed in a given temporal pattern.
  - a list (output `List.Glossary`) containing the glossary of the column names of `DE.results`.
  - a list (output `Summary.Inputs`) containing a summary of sample information and inputs of `DEanalysisGlobal()`.
- If data belong to different time points and different biological conditions (see `DEanalysisTimeAndGroup()`), the function returns
    - a data.frame (output `rowData(SEres)`) which contains
      - \* gene names
      - \* Results from the temporal statistical analysis
        - pvalues, log2 fold change and DE genes between each pairs of biological conditions for each fixed time.
        - $N_{bc}$  binary columns (0 and 1), one per biological condition (with  $N_{bc}$  the number of biological conditions). A 1 in one of these two columns means the gene is DE at least between one time  $t_i$  versus the reference time  $t_0$ , for the biological condition associated to the given column.
        - $N_{bc}$  columns, one per biological condition, where each element is succession of 0 and 1. The positions of 1 in one of these two columns, indicate the set of times  $t_i$  such that the gene is DE between  $t_i$  and the reference time  $t_0$ , for the biological condition associated to the given column.
      - \* Results from the statistical analysis by biological condition
        - pvalues, log2 fold change and DE genes between each time  $t_i$  and the reference time  $t_0$  for each biological condition.
        - $T$  binary columns (0 and 1), one per time (with  $T$  the number of time measurements). A 1 in one of these columns, means the gene is DE between at least one pair of biological conditions, for the fixed time associated to the given column.
        - $T \times N_{bc}$  binary columns, which give the genes specific for each biological condition at each time  $t_i$ . A 1 in one of these columns means the gene is specific to the biological condition at a fixed time associated to the given column. 0 otherwise. A gene is called specific to a given biological condition BC1 at a time  $t_i$ , if the gene is DE between BC1 and any other biological conditions at time  $t_i$ , but not DE between any pair of other biological conditions at time  $t_i$ .
        - $T \times N_{bc}$  columns filled with -1, 0 and 1. A 1 in one of these columns means the gene is up-regulated (or over-expressed) for the biological condition at a fixed time associated to the given column. A gene is called up-regulated for a given biological condition BC1 at time  $t_i$  if the gene is specific to the biological condition BC1 at time  $t_i$  and expressions in BC1 at time  $t_i$  are higher than in the other biological conditions at time  $t_i$ . A -1 in one of these columns means the gene is down-regulated

- (or under-expressed) for the biological condition at a fixed time associated to the given column. A gene is called down-regulated for a given biological condition at a time  $t_i$  BC1 if the gene is specific to the biological condition BC1 at time  $t_i$  and expressions in BC1 at time  $t_i$  are lower than in the other biological conditions at time  $t_i$ . A 0 in one of these columns means the gene is not specific to the biological condition at a fixed time associated to the given column.
- $N_{bc}$  binary columns (0 and 1). A 1 in one of these columns means the gene is specific at at least one time  $t_i$ , for the biological condition associated to the given column. 0 otherwise.
  - \* Results from the combination of temporal and biological statistical analysis
    - $T \times N_{bc}$  binary columns, which give the signatures genes for each biological condition at each time  $t_i$ . A 1 in one of these columns means the gene is signature gene to the biological condition at a fixed time associated to the given column. 0 otherwise. A gene is called signature of a biological condition BC1 at a given time  $t_i$ , if the gene is specific to the biological condition BC1 at time  $t_i$  and DE between  $t_i$  versus the reference time  $t_0$  for the biological condition BC1.
    - $N_{bc}$  binary columns (0 and 1). A 1 in one of these columns means the gene is signature at at least one time  $t_i$ , for the biological condition associated to the given column. 0 otherwise.
  - the following plots from the temporal statistical analysis
    - \* a barplot which gives the number of DE genes between  $t_i$  and the reference time  $t_0$ , for each time  $t_i$  (except the reference time  $t_0$ ) and biological condition (see [DEplotBarplotFacetGrid\(\)](#)).
    - \*  $N_{bc}$  alluvial graphs of DE genes (see [DEplotAlluvial\(\)](#)), one per biological condition.
    - \*  $N_{bc}$  graphs showing the number of DE genes as a function of time for each temporal group, one per biological condition. By temporal group, we mean the sets of genes which are first DE at the same time.
    - \*  $2 \times N_{bc}$  UpSet plot showing the number of DE genes belonging to each DE temporal pattern, for each biological condition. By temporal pattern, we mean the set of times  $t_i$  such that the gene is DE between  $t_i$  and the reference time  $t_0$  (see [DEplotVennBarplotTime\(\)](#)).
    - \* an alluvial graph for DE genes which are DE at least one time for each group.
  - the following plots from the statistical analysis by biological condition
    - \* a barplot which gives the number of specific DE genes for each biological condition and time (see [DEplotBarplotFacetGrid\(\)](#)).
    - \*  $N_{bc}(N_{bc}-1)/2$  UpSet plot which give the number of genes for each possible intersection (set of pairs of biological conditions), one per time (see [DEplotVennBarplotGroup\(\)](#)).
    - \* an alluvial graph of genes which are specific at least one time (see [DEplotAlluvial\(\)](#)).
  - the following plots from the combination of temporal and biological statistical analysis
    - \* a barplot which gives the number of signature genes for each biological condition and time (see [DEplotBarplotFacetGrid\(\)](#)).
    - \* a barplot showing the number of genes which are DE at at least one time, specific at at least one time and signature at at least one time, for each biological condition.
    - \* an alluvial graph of genes which are signature at least one time (see [DEplotAlluvial\(\)](#)).

## Examples

```

data(RawCounts_Antoszewski2022_MOUSEsub500)
## No time points. We take only two groups for the speed of the example
RawCounts_T1Wt <- RawCounts_Antoszewski2022_MOUSEsub500[seq_len(200),
                                                         seq_len(7)]

##-----##
## Preprocessing
resDATAprepSE <- DATAprepSE(RawCounts=RawCounts_T1Wt,
                             Column.gene=1,
                             Group.position=1,
                             Time.position=NULL,
                             Individual.position=2)

##-----##
## DE analysis
resDE <- DEanalysisGlobal(Seres=resDATAprepSE,
                          pval.min=0.05,
                          pval.vect.t=NULL,
                          log.FC.min=1,
                          LRT.suppl.info=FALSE,
                          Plot.DE.graph=TRUE,
                          path.result=NULL,
                          Name.folder.DE=NULL)

```

---

DEanalysisGroup

*DE Analysis when samples belong to different biological conditions.*


---

## Description

The function realizes from the `DESeq2::DESeq()` output the analysis of DE genes between all pairs of biological conditions.

## Usage

```

DEanalysisGroup(
  DESeq.result,
  pval.min = 0.05,
  log.FC.min = 1,
  LRT.suppl.info = TRUE,
  Plot.DE.graph = TRUE,
  path.result = NULL,
  SubFile.name = NULL
)

```

## Arguments

`DESeq.result` Output from the function `DESeq2::DESeq()`.

<code>pval.min</code>	Numeric value between 0 and 1. A gene will be considered as differentially expressed (DE) between two biological conditions if its Benjamini-Hochberg adjusted p-value (see <code>stats::p.adjust()</code> ) is below the threshold <code>pval.min</code> . Default value is 0.05.
<code>log.FC.min</code>	Non negative numeric value. If the log2 fold change between biological conditions or times has an absolute value below the threshold <code>log.FC.min</code> , then the gene is not selected even if is considered as DE. Default value is 1. If <code>log.FC.min=0</code> , all DE genes will be kept.
<code>LRT.suppl.info</code>	TRUE or FALSE. If TRUE, the algorithm realizes another statistical test in order to detect if, among all biological conditions and/or times, at least one has a different behavior than the others (see the input 'test' in <code>DESeq2::DESeq()</code> ).
<code>Plot.DE.graph</code>	TRUE or FALSE. TRUE as default. If TRUE, all graphs will be plotted. Otherwise no graph will be plotted.
<code>path.result</code>	Character or NULL. If <code>path.result</code> is a character, it must be a path to a folder, all graphs will be saved in <code>path.result</code> . If NULL, the results will not be saved in a folder. NULL as default.
<code>SubFile.name</code>	Character or NULL. If <code>SubFile.name</code> is a character, each saved file names will contain the strings of characters " <code>_SubFile.name</code> ". If NULL, no suffix will be added.

## Value

The function returns the same `DESeqDataSet` class object `DESeq.result` with the following results, saved in the metadata `DEresultsGroup` of `DESeq.result`:

- a `data.frame` (output `DEsummary` of `DEresultsGroup`) which contains
  - gene names
  - pvalues, log2 fold change and DE genes between each pairs of biological conditions.
  - a binary column (1 and 0) where 1 means the gene is DE between at least one pair of biological conditions.
  - $N_{bc}$  binary columns, where  $N_{bc}$  is the number of biological conditions, which gives the specific genes for each biological condition. A '1' in one of these columns means the gene is specific to the biological condition associated to the given column. 0 otherwise. A gene is called specific to a given biological condition BC1, if the gene is DE between BC1 and any other biological conditions, but not DE between any pair of other biological conditions.
  - $N_{bc}$  columns filled with -1, 0 and 1, one per biological condition. A '1' in one of these columns means the gene is up-regulated (or over-expressed) for the biological condition associated to the given column. A gene is called up-regulated for a given biological condition BC1 if the gene is specific to the biological condition BC1 and expressions in BC1 are higher than in the other biological conditions. A '-1' in one of these columns means the gene is down-regulated (or under-expressed) for the biological condition associated to the given column. A gene is called down-regulated for a given biological condition BC1 if the gene is specific to the biological condition BC1 and expressions in BC1 are lower than in the other biological conditions. A '0' in one of these columns means the gene is not specific to the biological condition associated to the given column.

- A contingency matrix (output Summary.DEanalysis of DEresultsGroup) which gives for each biological condition the number of genes categorized as "Upregulated", "DownRugulated" and "Other". A gene is categorized as 'Other', for a given biological condition, if the gene is not specific to the given biological condition. The category 'Other' does not exist when there are only two biological conditions.
- an UpSet plot (Venn diagram displayed as a barplot) which gives the number of genes for each possible intersection (see `DEplotVennBarplotGroup()`). We consider that a set of pairs of biological conditions forms an intersection if there is at least one gene which is DE for each of these pairs of biological conditions, but not for the others.
- a barplot which gives the number of genes categorized as "Upregulated" and "DownRugulated", per biological condition (see `DEplotBarplot()`).
- a barplot which gives the number of genes categorized as "Upregulated", "DownRugulated" and "Other", per biological condition (see `DEplotBarplot()`). So this barplot, only plotted when there are strictly more than two biological conditions, is similar to the previous barplot but with the category "Other".

### See Also

The outputs of the function are used by the main function `DEanalysisGlobal()`.

### Examples

```
## Data
data(RawCounts_Antoszewski2022_MOUSEsub500)
## No time points. We take only two groups for the speed of the example
RawCounts_T1Wt<-RawCounts_Antoszewski2022_MOUSEsub500[seq_len(200),
                                                    seq_len(7)]

## Preprocessing step
resDATAprepSEmus1<- DATAprepSE(RawCounts=RawCounts_T1Wt,
                                Column.gene=1,
                                Group.position=1,
                                Time.position=NULL,
                                Individual.position=2)

DESeq2preprocess <- S4Vectors::metadata(resDATAprepSEmus1)$DESeq2obj
DESeq2obj <- DESeq2preprocess$DESeq2preproceesing

##-----##
dds.DE.G <- DESeq2::DESeq(DESeq2obj, quiet=TRUE, betaPrior=FALSE)

res.sum.group <- DEanalysisGroup(DESeq.result=dds.DE.G,
                                pval.min=0.01,
                                log.FC.min=1,
                                LRT.suppl.info=FALSE,
                                Plot.DE.graph=TRUE,
                                path.result=NULL,
                                SubFile.name=NULL)
```



---

DEanalysisSubData      *Sub data of a data.frame*


---

### Description

From the results from our function `DEanalysisGlobal()`, the function extracts from the `SummarizedExperiment` class outputs of the subset of genes selected with the inputs `Set.Operation` and `ColumnsCriteria`, and saves them in a `SummarizeExperiment` object.

### Usage

```
DEanalysisSubData(
  SEresDE,
  ColumnsCriteria = 1,
  Set.Operation = "union",
  Save.SubData = FALSE
)
```

### Arguments

<code>SEresDE</code>	A <code>SummarizedExperiment</code> class object. Output from <code>DEanalysisGlobal()</code> (see Examples).
<code>ColumnsCriteria</code>	A vector of integers where each integer indicates a column of <code>SummarizedExperiment::rowData(SEresDE)</code> . These columns should either contain only binary values, or may contain other numerical value, in which case extracted outputs from <code>SEresDE</code> will be those with $>0$ values (see Details).
<code>Set.Operation</code>	A character. The user must choose between "union" (default), "intersect", "set-diff" (see Details).
<code>Save.SubData</code>	TRUE or FALSE or a Character. FALSE as default. If TRUE, two csv files (see Value) will be saved in the folder "2_SupervisedAnalysis_Name.folder.DE" (see <code>DEanalysisGlobal()</code> ).

### Details

We have the following three cases:

- If `Set.Operation="union"` then the rows extracted from the different datasets included in `SEresDE` are those such that the sum of the selected columns of `SummarizedExperiment::rowData(SEresDE)` by `ColumnsCriteria` is  $>0$ . For example, the selected genes can be those DE at least at  $t_1$  or  $t_2$  (versus the reference time  $t_0$ ).
- If `Set.Operation="intersect"` then the rows extracted from the different datasets included in `SEresDE` are those such that the product of the selected columns of `SummarizedExperiment::rowData(SEresDE)` by `ColumnsCriteria` is  $>0$ . For example, the selected genes can be those DE at times  $t_1$  and  $t_2$  (versus the reference time  $t_0$ ).

- If `Set.Operation="setdiff"` then the rows extracted from the different datasets included in `SeresDE` are those such that only one element of the selected columns of `SummarizedExperiment::rowData(SeresDE)` by `ColumnsCriteria` is  $>0$ . For example, the selected genes can be those DE at times `t1` only and at times `t2` only (versus the reference time `t0`).

### Value

The function returns a `SummarizeExperiment` class object containing

- sub data.frames of the different dataset included in `SeresDE` containing only the rows specified by `ColumnsCriteria` and `Set.Operation`.
- the DE results saved in `SeresDE` of genes selected by `ColumnsCriteria` and `Set.Operation`.
- The genes specified by `ColumnsCriteria` and `Set.Operation`.

### Examples

```
## Simulation raw counts
resSIMcount <- RawCountsSimulation(Nb.Group=2, Nb.Time=1, Nb.per.GT=4,
                                   Nb.Gene=5)

## Preprocessing step
resDATAprepSE <- DATAprepSE(RawCounts=resSIMcount$Sim.dat,
                              Column.gene=1,
                              Group.position=1,
                              Time.position=NULL,
                              Individual.position=2)

##-----##
## Transformation of resDATAprepSE into results of DEanalysisGlobal
resultsExamples <- data.frame(Gene=paste0("Gene", seq_len(5)),
                              DE1=c(0, 1, 0, 0, 1),
                              DE2=c(0, 1, 0, 1, 0))
listPATHnameEx <- list(Path.result=NULL, Folder.result=NULL)

SummarizedExperiment::rowData(resDATAprepSE) <- resultsExamples
S4Vectors::metadata(resDATAprepSE)$DESeq2obj$pathNAME <- listPATHnameEx
S4Vectors::metadata(resDATAprepSE)$DESeq2obj$SEidentification<-"SeresultsDE"

##-----##
## results of DEanalysisSubData
resDEsub <- DEanalysisSubData(SeresDE=resDATAprepSE,
                              ColumnsCriteria=c(2, 3),
                              Set.Operation="union",
                              Save.SubData=FALSE)
```

## Description

The function realizes from the `DESeq2::DESeq()` output the analysis of DE genes between each time versus the reference time  $t_0$ .

## Usage

```
DEanalysisTime(
  DESeq.result,
  pval.min = 0.05,
  pval.vect.t = NULL,
  log.FC.min = 1,
  LRT.suppl.info = FALSE,
  Plot.DE.graph = TRUE,
  path.result = NULL,
  SubFile.name = NULL
)
```

## Arguments

<code>DESeq.result</code>	Output from the function <code>DESeq2::DESeq()</code> .
<code>pval.min</code>	Numeric value between 0 and 1. A gene will be considered as differentially expressed (DE) between two biological conditions if its Benjamini-Hochberg adjusted p-value (see <code>stats::p.adjust()</code> ) is below the threshold <code>pval.min</code> . Default value is 0.05
<code>pval.vect.t</code>	NULL or vector of dimension $T - 1$ filled with numeric values between 0 and 1, with $T$ the number of time measurements. A gene will be considered as differentially expressed (DE) between the time $t_i$ and the reference time $t_0$ if its Benjamini-Hochberg adjusted p-value (see <code>stats::p.adjust()</code> ) is below the $i$ -th threshold of <code>pval.vect.t</code> . If NULL, <code>pval.vect.t</code> will be vector of dimension $T - 1$ filled with <code>pval.min</code> .
<code>log.FC.min</code>	Non negative numeric value. If the $\log_2$ fold change between biological conditions or times has an absolute value below the threshold <code>log.FC.min</code> , then the gene is not selected even if is considered as DE. Default value is 1. If <code>log.FC.min=0</code> , all DE genes will be kept.
<code>LRT.suppl.info</code>	TRUE or FALSE. If TRUE, the algorithm realizes another statistical test in order to detect if, among all biological conditions and/or times, at least one has a different behavior than the others (see the input 'test' in <code>DESeq2::DESeq()</code> ).
<code>Plot.DE.graph</code>	TRUE or FALSE. TRUE as default. If TRUE, all graphs will be plotted. Otherwise no graph will be plotted.
<code>path.result</code>	Character or NULL. If <code>path.result</code> is a character, it must be a path to a folder, all graphs will be saved in <code>path.result</code> . If NULL, the results will not be saved in a folder. NULL as default.
<code>SubFile.name</code>	Character or NULL. If <code>SubFile.name</code> is a character, each saved file names will contain the strings of characters " <code>_SubFile.name</code> ". If NULL, no suffix will be added.

**Value**

The function returns the same `DESeqDataSet` class object `DESeq.result` with the following results, saved in the metadata `DEresultsTime` of `DESeq.result`:

- a data.frame (output `DEsummary` of `DEresultsTime`) which contains
  - gene names
  - pvalues, log<sub>2</sub> fold change and DE genes between each time `ti` versus the reference time `t0`.
  - a binary column (1 and 0) where 1 means the gene is DE at least at between one time `ti` versus the reference time `t0`.
  - a column where each element is succession of 0 and 1. The positions of '1' indicate the set of times `ti` such that the gene is DE between `ti` and the reference time `t0`.
- an alluvial graph of differentially expressed (DE) genes (see `DEplotAlluvial()`)
- a graph showing the number of DE genes as a function of time for each temporal group (see `DEplotAlluvial()`). By temporal group, we mean the sets of genes which are first DE at the same time.
- a barplot which gives the number of DE genes per time (see `DEplotBarplotTime()`)
- an UpSet plot (Venn diagram displayed as a barplot) which gives the number of genes per temporal pattern (see `DEplotVennBarplotTime()`). By temporal pattern, we mean the set of times `ti` such that the gene is DE between `ti` and the reference time `t0`.
- a similar UpSet plot where each bar is split in different colors corresponding to all possible numbers of DE times where genes are over expressed in a given temporal pattern.

**See Also**

The outputs of the function will be used by the main function `DEanalysisGlobal()`.

**Examples**

```
data(RawCounts_Leong2014_FISSIONsub500wt)
## We take only the first three time for the speed of the example
RawCounts_Fission_3t<-RawCounts_Leong2014_FISSIONsub500wt[seq_len(200),
                                                         seq_len(10)]

## Preprocessing step
resDATAprepSEfission <- DATAprepSE(RawCounts=RawCounts_Fission_3t,
                                     Column.gene=1,
                                     Group.position=NULL,
                                     Time.position=2,
                                     Individual.position=3)

DESeq2preprocess <- S4Vectors::metadata(resDATAprepSEfission)$DESeq2obj
DESeq2obj <- DESeq2preprocess$DESeq2preproceesing

##-----##
dds.DE.T <- DESeq2::DESeq(DESeq2obj, quiet=TRUE, betaPrior=FALSE)
##
res.T <- DEanalysisTime(DESeq.result=dds.DE.T,
```

```
pval.min=0.05,
pval.vect.t=c(0.01,0.05,0.05),
log.FC.min=1,
LRT.suppl.info=FALSE,
Plot.DE.graph=TRUE,
path.result=NULL,
SubFile.name=NULL)
```

---

DEanalysisTimeAndGroup

*DE analysis when samples belong to different biological condition and time points.*

---

## Description

The function realizes from the [DESeq2::DESeq\(\)](#) output the analysis of :

- DE genes between all pairs of biological conditions for each fixed time.
- DE genes between all times  $t_i$  and the reference time  $t_0$  for each biological condition.

## Usage

```
DEanalysisTimeAndGroup(
  DESeq.result,
  LRT.suppl.info = TRUE,
  log.FC.min,
  pval.min,
  pval.vect.t,
  Plot.DE.graph = TRUE,
  path.result,
  SubFile.name
)
```

## Arguments

DESeq.result	Output from the function <a href="#">DESeq2::DESeq()</a> .
LRT.suppl.info	TRUE or FALSE. If TRUE, the algorithm realizes another statistical test in order to detect if, among all biological conditions and/or times, at least one has a different behavior than the others (see the input test in <a href="#">DESeq2::DESeq()</a> ).
log.FC.min	Non negative numeric value. If the $\log_2$ fold change between biological conditions or times has an absolute value below the threshold $\log.FC.min$ , then the gene is not selected even if is considered as DE. Default value is 1. If $\log.FC.min=0$ , all DE genes will be kept.
pval.min	Numeric value between 0 and 1. A gene will be considered as differentially expressed (DE) between two biological conditions if its Benjamini-Hochberg adjusted p-value (see <a href="#">stats::p.adjust()</a> ) is below the threshold $pval.min$ . Default value is 0.05

<code>pval.vect.t</code>	NULL or vector of dimension $T - 1$ filled with numeric values between 0 and 1, with $T$ the number of time measurements. A gene will be considered as differentially expressed (DE) between the time $t_i$ and the reference time $t_0$ if its Benjamini-Hochberg adjusted p-value (see <code>stats::p.adjust()</code> ) is below the $i$ -th threshold of <code>pval.vect.t</code> . If NULL, <code>pval.vect.t</code> will be vector of dimension $T - 1$ filled with <code>pval.min</code> .
<code>Plot.DE.graph</code>	TRUE or FALSE. TRUE as default. If TRUE, all graphs will be plotted. Otherwise no graph will be plotted.
<code>path.result</code>	Character or NULL. If <code>path.result</code> is a character, it must be a path to a folder, all graphs will be saved in different sub-folders in <code>path.result</code> . If NULL, the results will not be saved. NULL as default.
<code>SubFile.name</code>	Character or NULL. If <code>SubFile.name</code> is a character, each saved file names and created folders names will contain the strings of characters " <code>_SubFile.name</code> ". If NULL, no suffix will be added.

## Value

The function returns the same `DESeqDataSet` class object `DESeq.result` with the following results, saved in the metadata `DEresultsTimeGroup` of `DESeq.result`:

- a `data.frame` (output `DEsummary` of `DEresultsTimeGroup`) which contains
  - gene names
  - Results from the temporal statistical analysis
    - \* pvalues, log2 fold change and DE genes between each pairs of biological conditions for each fixed time.
    - \*  $N_{bc}$  binary columns (0 and 1), one per biological condition (with  $N_{bc}$  the number of biological conditions). A 1 in one of these two columns means the gene is DE at least between one time  $t_i$  versus the reference time  $t_0$ , for the biological condition associated to the given column.
    - \*  $N_{bc}$  columns, one per biological condition, where each element is succession of 0 and 1. The positions of 1 in one of these two columns, indicate the set of times  $t_i$  such that the gene is DE between  $t_i$  and the reference time  $t_0$ , for the biological condition associated to the given column.
  - Results from the statistical analysis by biological condition
    - \* pvalues, log2 fold change and DE genes between each time  $t_i$  and the reference time  $t_0$  for each biological condition.
    - \*  $T$  binary columns (0 and 1), one per time (with  $T$  the number of time measurements). A 1 in one of these columns, means the gene is DE between at least one pair of biological conditions, for the fixed time associated to the given column.
    - \*  $T \times N_{bc}$  binary columns, which give the genes specific for each biological condition at each time  $t_i$ . A 1 in one of these columns means the gene is specific to the biological condition at a fixed time associated to the given column. 0 otherwise. A gene is called specific to a given biological condition BC1 at a time  $t_i$ , if the gene is DE between BC1 and any other biological conditions at time  $t_i$ , but not DE between any pair of other biological conditions at time  $t_i$ .

- \*  $T \times N_{bc}$  columns filled with -1, 0 and 1. A 1 in one of these columns means the gene is up-regulated (or over-expressed) for the biological condition at a fixed time associated to the given column. A gene is called up-regulated for a given biological condition BC1 at time  $t_i$  if the gene is specific to the biological condition BC1 at time  $t_i$  and expressions in BC1 at time  $t_i$  are higher than in the other biological conditions at time  $t_i$ . A -1 in one of these columns means the gene is down-regulated (or under-expressed) for the biological condition at a fixed time associated to the given column. A gene is called down-regulated for a given biological condition at a time  $t_i$  BC1 if the gene is specific to the biological condition BC1 at time  $t_i$  and expressions in BC1 at time  $t_i$  are lower than in the other biological conditions at time  $t_i$ . A 0 in one of these columns means the gene is not specific to the biological condition at a fixed time associated to the given column.
- \*  $N_{bc}$  binary columns (0 and 1). A 1 in one of these columns means the gene is specific at least at one time  $t_i$ , for the biological condition associated to the given column. 0 otherwise.
- Results from the combination of temporal and biological statistical analysis
  - \*  $T \times N_{bc}$  binary columns, which give the signatures genes for each biological condition at each time  $t_i$ . A 1 in one of these columns means the gene is signature gene to the biological condition at a fixed time associated to the given column. 0 otherwise. A gene is called signature of a biological condition BC1 at a given time  $t_i$ , if the gene is specific to the biological condition BC1 at time  $t_i$  and DE between  $t_i$  versus the reference time  $t_0$  for the biological condition BC1.
  - \*  $N_{bc}$  binary columns (0 and 1). A 1 in one of these columns means the gene is signature at least at one time  $t_i$ , for the biological condition associated to the given column. 0 otherwise.
- the following plots from the temporal statistical analysis
  - a barplot which gives the number of DE genes between  $t_i$  and the reference time  $t_0$ , for each time  $t_i$  (except the reference time  $t_0$ ) and biological condition (see [DEplotBarplotFacetGrid\(\)](#)).
  - $N_{bc}$  alluvial graphs of DE genes (see [DEplotAlluvial\(\)](#)), one per biological condition.
  - $N_{bc}$  graphs showing the number of DE genes as a function of time for each temporal group, one per biological condition. By temporal group, we mean the sets of genes which are first DE at the same time.
  - $2 \times N_{bc}$  UpSet plot showing the number of DE genes belonging to each DE temporal pattern, for each biological condition. By temporal pattern, we mean the set of times  $t_i$  such that the gene is DE between  $t_i$  and the reference time  $t_0$  (see [DEplotVennBarplotTime\(\)](#)).
  - an alluvial graph for DE genes which are DE at least one time for each group.
- the following plots from the statistical analysis by biological condition
  - a barplot which gives the number of specific DE genes for each biological condition and time (see [DEplotBarplotFacetGrid\(\)](#)).
  - $N_{bc}(N_{bc} - 1)/2$  UpSet plot which give the number of genes for each possible intersection (set of pairs of biological conditions), one per time (see [DEplotVennBarplotGroup\(\)](#)).
  - an alluvial graph of genes which are specific at least one time (see [DEplotAlluvial\(\)](#)).
- the following plots from the combination of temporal and biological statistical analysis
  - a barplot which gives the number of signature genes for each biological condition and time (see [DEplotBarplotFacetGrid\(\)](#)).

- a barplot showing the number of genes which are DE at least at one time, specific at least at one time and signature at least at one time, for each biological condition.
- an alluvial graph of genes which are signature at least one time (see `DEplotAlluvial()`).

### See Also

The outputs of the function will be used by the main function `DEanalysisGlobal()`.

### Examples

```
data(RawCounts_Schleiss2021_CLLsub500)
## We take only the first three times (both group) for the speed of
## the example
Index3t<-c(2:4,11:13,20:22, 29:31,38:40,47:49)
RawCounts_3t<-RawCounts_Schleiss2021_CLLsub500[seq_len(200), c(1,Index3t)]

## Preprocessing step
resDATAprepSEleuk <- DATAprepSE(RawCounts=RawCounts_3t,
                                 Column.gene=1,
                                 Group.position=2,
                                 Time.position=4,
                                 Individual.position=3)

DESeq2preprocess <- S4Vectors::metadata(resDATAprepSEleuk)$DESeq2obj
DESeq2obj <- DESeq2preprocess$DESeq2preprocessing

##-----##
dds.DE <- DESeq2::DESeq(DESeq2obj)
##
res.G.T <- DEanalysisTimeAndGroup(DESeq.result=dds.DE,
                                 LRT.suppl.info=FALSE,
                                 pval.min=0.05,
                                 pval.vect.t=NULL,
                                 log.FC.min=0.1,
                                 Plot.DE.graph=TRUE,
                                 path.result=NULL,
                                 SubFile.name=NULL)
```

---

DEplotAlluvial

*Alluvial graphs of differentially expressed (DE) genes*

---

### Description

The function takes as input a binary table with  $N_g$  lines corresponding to genes and

- if `Temporal.Group=TRUE` :  $T - 1$  columns corresponding to times (with  $T$  the number of time points). A '1' in the  $n$ -th row and  $t$ -th column means that the  $n$ -th gene is differentially expressed (DE) at time  $t$ , compared with the reference time  $t_0$ .
- if `Temporal.Group=FALSE` :  $G$  columns corresponding to the number of group. A '1' in the  $n$ -th row and  $g$ -th column means that the  $n$ -th gene is



- DE at least one time  $t_i$ , compared with the reference time  $t_0$ , for the group  $g$ .
- specific at least one time  $t_i$ , compared with the reference time  $t_0$ , for the group  $g$  (see [DEanalysisTimeAndGroup\(\)](#) for the notion 'specific').
- a signature gene at least one time  $t_i$ , compared with the reference time  $t_0$ , for the group  $g$  (see [DEanalysisTimeAndGroup\(\)](#) for the notion 'signature').

The function plots

- if `Temporal.Group=TRUE`, two graphs: an alluvial graph and a plot showing the time evolution of the number of DE genes within each temporal group. By temporal group, we mean the sets of genes which are first DE at the same time.
- if `Temporal.Group=FALSE` : an alluvial graph.

## Usage

```
DEplotAlluvial(
  table.DE.time,
  Temporal.Group = TRUE,
  title.alluvial = NULL,
  title.evolution = NULL
)
```

## Arguments

`table.DE.time` Binary matrix (table filled with 0 and 1) with  $N_g$  rows and  $T - 1$  columns with  $N_g$  the number of genes and  $T - 1$  the number of time points.

`Temporal.Group` TRUE or FALSE, FALSE as default (see Description).

`title.alluvial` String of characters or NULL, NULL as default. The input `title.alluvial` corresponds to the title of the alluvial graph. If `title` is a string of characters, `title` will be the title of the alluvial graph. If `title=NULL`, the title of the alluvial graph will be 'Alluvial graph'.

`title.evolution` String of characters or NULL, NULL as default. Only applied if `Temporal.Group=TRUE`. The input `title.evolution` corresponds to the title of the second graph (see Description). If `title` is a string of characters, it will be to the title of the second graph. If `title=NULL`, the title of the second graph will be 'Time evolution of the number of DE genes within each temporal group'.

## Details

The names of the columns of the table will be the axis labels in the plots. If the table has no column names, the function will automatically create column names ( $t_1, t_2, \dots$ ).

## Value

The function returns, as described in description

- if `Temporal.Group=TRUE`, two graphs: an alluvial graph and a plot showing the time evolution of the number of DE genes within each temporal group. By temporal group, we mean the sets of genes which are first DE at the same time.

- if `Temporal.Group=FALSE` : an alluvial graph.

### See Also

The `DEplotAlluvial()` function

- is used by the following functions of our package : `DEanalysisTime()` and `DEanalysisTimeAndGroup()`.
- calls the R package `ggplot2` in order to plot the two graphs.

### Examples

```
set.seed(1994)

NbTime.vst0 <- 4
BinTable <- matrix(sample(c(0,1),replace=TRUE,
                          size=NbTime.vst0*120,c(0.60,0.40)),
                   ncol=NbTime.vst0)
colnames(BinTable) <- paste0("t", 1:NbTime.vst0)

##-----##
res.alluvial <- DEplotAlluvial(table.DE.time=BinTable)
print(res.alluvial$g.alluvial)
print(res.alluvial$g.alluvial.freq)
```

---

DEplotBarplot

*Barplot of DE genes from a contingency table.*

---

### Description

From a contingency table between two variables, the function plots a barplot of the frequency distribution of one variable against the other (see Details).

### Usage

```
DEplotBarplot(ContingencyTable, dodge = TRUE)
```

### Arguments

ContingencyTable

A numeric data.frame, corresponding to a contingency table, of dimension  $N1 \times N2$ , with  $N1$  and  $N2$ , respectively the number of levels in the first and second variable (see examples and details).

dodge

TRUE or FALSE. FALSE means multiple bars in the barplot (one per level of the first variable) one for each fixed level of the other variable. TRUE means multiple bars will be dodged side-to-side (see `ggplot2::geom_bar()`).

## Details

A contingency table (or cross-tabulation) is a table that displays the frequency distribution of two variables (each containing several levels), i.e. the number of observation recorded per pair of levels. The function plots a single barplot from `ContingencyTable`.

This function is called by `DEanalysisGroup()` and `DEanalysisTimeAndGroup()`. These two functions produce several contingency tables, giving information about specific and particular DE genes, as described below.

First, we look for all genes that are DE between at least two biological conditions. A gene will be called specific to a given biological condition BC1, if the gene is DE between BC1 and any other biological conditions, but not DE between any pair of other biological conditions. Then each DE gene will be categorized as follow:

- If a gene is not specific, the gene will be categorized as 'Other'. The category 'Other' does not exist when there are only two biological conditions.
- If a gene is specific to a given biological condition BC1 and expressions in BC1 are higher than in the other biological conditions, the gene will be categorized as 'Upregulated'.
- If a gene is specific to a given biological condition BC1 and expressions in BC1 are lower than in the other biological conditions, the gene will be categorized as 'Downregulated'.

The functions `DEanalysisGroup()` and `DEanalysisTimeAndGroup()` produce two contingency table that allow to plot both

- the number of genes categorized as 'Other', 'Upregulated' and 'Downregulated' (only when there are strictly more than two biological conditions).
- the number of genes categorized 'Upregulated' and 'Downregulated'.

Second, we look for all genes that are DE between at least one time point (except t0) and t0 for each biological condition. A gene will be categorized as 'particular' to a given biological condition BC1 for a given time point ti (except t0), if the gene is DE between ti and t0 for the biological condition BC1, but not DE between ti and t0 for the other biological conditions. A gene will be categorized as 'common' to all biological conditions, if the gene is DE between ti and t0 for all biological conditions. Otherwise, a gene will categorized as 'Other'.

The function `DEanalysisTimeAndGroup()` produces a contingency table that allow to plot the number of 'specific', 'common' and 'other' genes for each ti (except t0).

## Value

A barplot using `ggplot2` (see details).

## See Also

The `DEplotBarplot()` function

- is used by the following functions of our package: `DEanalysisGroup()` and `DEanalysisTimeAndGroup()`.
- calls the R package `ggplot2` in order to plot the barplot.

**Examples**

```
## Data simulation
CrossTabulation <- matrix(c(75,30,10,5, 5,35,5,20, 220,235,285,275),
                          ncol=4, byrow=TRUE)
colnames(CrossTabulation) <- c("A", "B", "C", "D")
row.names(CrossTabulation) <- c("Spe.Pos", "Spe.Neg", "Other")

##-----##
res.dodgeTRUE <- DEplotBarplot(ContingencyTable=CrossTabulation,dodge=FALSE)
res.dodgeTRUE

res.dodgeFALSE <- DEplotBarplot(ContingencyTable=CrossTabulation,dodge=TRUE)
res.dodgeFALSE
```

---

DEplotBarplotFacetGrid

*Faceted barplot of specific DE genes*


---

**Description**

The function creates a faceted barplot from a data.frame containing two or three qualitative variables and one quantitative variable.

**Usage**

```
DEplotBarplotFacetGrid(
  Data,
  Abs.col,
  Legend.col,
  Facet.col,
  Value.col,
  Color.Legend = NULL,
  LabsPlot = c("", "")
)
```

**Arguments**

Data	Data.frame containing three or four columns. One must contain quantitative variable and the other qualitative variables.
Abs.col	Integer indicating the column of Data which will be used for the x-axis. The selected column must be one of the qualitative variables and must be identical to Legend.col if there are only two qualitative variables. Otherwise, Abs.col and Legend.col must be different.
Legend.col	Integer indicating the column of Data which is used for the color of the barplots. The selected column must be one of the qualitative variables and must be identical to Abs.col if there are only two qualitative variables. Otherwise, Abs.col and Legend.col must be different.

Facet.col	Integer indicating the column of Data which is used for separating barplots in different panels, one per level of the qualitative variable. The selected column must be one of the qualitative variables.
Value.col	Integer indicating the column of Data which contains numeric values.
Color.Legend	Data.frame or NULL. If Color.Legend is a data.frame, the data.frame must have two columns and $N_{bc}$ rows where $N_{bc}$ is the number of biological conditions. The first column must contain the name of the $N_{bc}$ different biological conditions and the second column must be the color associated to each biological condition. If Color.Legend=NULL, the function will automatically attribute a color for each biological condition.
LabsPlot	Vector of two characters indicating the x-axis label and the y-axis label of the facet grid barplot. By default, LabsPlot=c("", "").

### Value

The function will plot a facet grid barplot. The function is called by our function [DEanalysisTimeAndGroup\(\)](#) in order to plot the number of specific (up- or down-regulated) DE genes per biological condition for each time points.

### See Also

The function

- is called by the function [DEanalysisTimeAndGroup\(\)](#)
- calls the R functions [ggplot2::facet\\_grid\(\)](#) and [ggplot2::geom\\_bar\(\)](#).

### Examples

```
Group.ex <- c('G1', 'G2', 'G3')
Time.ex <- c('t1', 't2', 't3', 't4')
Spe.sign.ex <- c("Pos", "Neg")
GtimesT <- length(Group.ex)*length(Time.ex)

Nb.Spe <- sample(3:60, GtimesT, replace=FALSE)
Nb.Spe.sign <- sample(3:60, 2*GtimesT, replace=FALSE)

##-----##
Melt.Dat.1 <- data.frame(Group=rep(Group.ex, times=length(Time.ex)),
                        Time=rep(Time.ex, each=length(Group.ex)),
                        Nb.Spe.DE=Nb.Spe)

DEplotBarplotFacetGrid(Data=Melt.Dat.1, Abs.col=2, Legend.col=2,
                       Facet.col=1, Value.col=3, Color.Legend=NULL)
DEplotBarplotFacetGrid(Data=Melt.Dat.1, Abs.col=1, Legend.col=1,
                       Facet.col=2, Value.col=3, Color.Legend=NULL)

##-----##
Melt.Dat.2 <- data.frame(Group=rep(Group.ex, times=length(Time.ex)*2),
                        Time=rep(Time.ex, each=length(Group.ex)*2),
                        Spe.sign=rep(Spe.sign.ex, times=2*GtimesT),
                        Nb.Spe.DE=Nb.Spe.sign)
```

```
DEplotBarplotFacetGrid(Data=Melt.Dat.2,
                        Abs.col=1,
                        Legend.col=3,
                        Facet.col=2,
                        Value.col=4,
                        Color.Legend=NULL)
```

---

DEplotBarplotTime      *Barplot of DE genes per time*

---

### Description

The function takes as input two tables

- a binary table with  $N_g$  rows corresponding to genes and  $T - 1$  columns corresponding to times (with  $T$  the number of time points). A '1' in the n-th row and i-th column means that the n-th gene is differentially expressed (DE) at time  $t_i$ , compared with the reference time  $t_0$ .
- a numeric matrix with positive and negative values with  $N_g$  rows corresponding to genes and  $T - 1$  columns corresponding to times. The element in n-th row and i-th column corresponds to the log2 fold change between the time  $t_i$  and the reference time  $t_0$  for the n-th gene. If the gene is DE and the sign is positive, then the gene n will be considered as over-expressed (up-regulated) at the time  $t_i$ . If the gene is DE and the sign is negative, then the gene n will be considered as under-expressed (down-regulated) at the time  $t_i$ .

The function plots two graphs: a barplot showing the number of DE genes per time and a barplot showing the number of under- and over-expressed genes per times.

### Usage

```
DEplotBarplotTime(table.DE.time, Log2.FC.matrix)
```

### Arguments

`table.DE.time` Binary matrix (table filled with 0 and 1) with  $N_g$  rows and  $T - 1$  columns with  $N_g$  the number of genes and  $T$  the number of time points.

`Log2.FC.matrix` Numeric matrix with positive and negative with  $N_g$  rows and  $T - 1$  columns.

### Value

The function plots two graphs: a barplot showing the number of DE genes per time and a barplot showing the number of under and over expressed genes per times.

**Examples**

```

set.seed(1994)
Dat1.FTP <- matrix(sample(c(0,1), replace=TRUE, size=120, prob=c(0.3,0.7)),
                   ncol=3)
Dat2.FTP <- matrix(round(rnorm(n=120, mean=0, sd=1),digits=2), ncol=3)
colnames(Dat1.FTP) <- paste0("t", 1:3)
colnames(Dat2.FTP) <- paste0("t", 1:3)
##-----###
res.DE.all.t <- DEplotBarplotTime(table.DE.time=Dat1.FTP,
                                Log2.FC.matrix=Dat2.FTP)
print(res.DE.all.t$g.nb.DEPerTime)
print(res.DE.all.t$g.nb.DEPerTime.sign)

```

DEplotHeatmaps

*Heatmaps of DE genes***Description**

The function returns two heatmaps: one heatmap of gene expressions between samples and selected genes and a correlation heatmap between samples from the output of [DEanalysisGlobal\(\)](#).

**Usage**

```

DEplotHeatmaps(
  SResDE,
  ColumnsCriteria = 2,
  Set.Operation = "union",
  NbGene.analysis = 20,
  Color.Group = NULL,
  SizeLabelRows = 5,
  SizeLabelCols = 5,
  Display.plots = TRUE,
  Save.plots = FALSE
)

```

**Arguments**

SResDE	A SummarizedExperiment class object. Output from <a href="#">DEanalysisGlobal()</a> (see Examples).
ColumnsCriteria	A vector of integers where each integer indicates a column of <code>SummarizedExperiment::rowData(SResDE)</code> . These columns should either contain only binary values, or may contain other numerical value, in which case extracted outputs from SResDE will be those with >0 values (see Details).
Set.Operation	A character. The user must choose between "union" (default), "intersect", "set-diff" (see Details).

<code>NbGene.analysis</code>	An integer or NULL. If it is an integer, the heatmaps will be plotted with the <code>NbGene.analysis</code> genes which have the highest sum of absolute log2 fold change, among the DE genes selected using <code>ColumnsCriteria</code> and <code>Set.Operation</code> . If NULL, all the DE selected genes will be used for both heatmaps.
<code>Color.Group</code>	NULL or a data.frame with $N_{bc}$ rows and two columns where $N_{bc}$ is the number of biological conditions. If <code>Color.Group</code> is a data.frame, the first column must contain the name of each biological condition and the second column must contain the colors associated to each biological condition. If <code>Color.Group=NULL</code> , the function will automatically attribute a color for each biological condition. If samples belong to different time points only, <code>Color.Group</code> will not be used.
<code>SizeLabelRows</code>	Numeric >0. Size of the labels for the genes in the heatmaps.
<code>SizeLabelCols</code>	Numeric >0. Size of the labels for the samples in the heatmaps.
<code>Display.plots</code>	TRUE or FALSE. TRUE as default. If TRUE, all graphs will be plotted. Otherwise no graph will be plotted.
<code>Save.plots</code>	TRUE or FALSE or a Character. If <code>Save.plots=FALSE</code> , the different files will not be saved. If <code>Save.plots=TRUE</code> and the path.result of <code>DEanalysisGlobal()</code> is not NULL, all files will be saved in "2_SupervisedAnalysis_Name.folder.DE/2-4_Supplementary_Plots_Name.folder.DE/Plots_Heatmaps". If <code>Save.plots</code> is a character, it must be a path and all files will be saved in the sub-folder "Plots_Heatmaps".

## Details

We have the following three cases:

- If `Set.Operation="union"` then the rows extracted from the different datasets (raw counts, normalized data and `SummarizedExperiment::rowData(SeresDE)`) included in the `SummarizedExperiment` class object `SeresDE` are those such that the sum of the selected columns of `SummarizedExperiment::rowData(SeresDE)` given in `ColumnsCriteria` is >0. This means that the selected genes are those having at least one '1' in one of the selected columns.
- If `Set.Operation="intersect"` then the rows extracted from the different datasets (raw counts, normalized data and `SummarizedExperiment::rowData(SeresDE)`) included in the `SummarizedExperiment` class object `SeresDE` are those such that the product of the selected columns of `SummarizedExperiment::rowData(SeresDE)` given in `ColumnsCriteria` is >0. This means that the selected genes are those having '1' in all of the selected columns.
- If `Set.Operation="setdiff"` then the rows extracted from the different datasets (raw counts, normalized data and `SummarizedExperiment::rowData(SeresDE)`) included in the `SummarizedExperiment` class object `SeresDE` are those such that only one element of the selected columns of `SummarizedExperiment::rowData(SeresDE)` given in `ColumnsCriteria` is >0. This means that the selected genes are those having '1' in only one of the selected columns.

## Value

The function returns the same `SummarizedExperiment` class object `SeresDE` with two heatmaps saved in the metadata `Results[[2]][[4]]` of `SeresDE`

- a correlation heatmap between samples (correlation heatmap)



- a heatmap across samples and genes called Zscore heatmap, for a subset of genes that can be selected by the user.

The two heatmaps are plotted if `Display.plots=TRUE`. The second heatmap is build from the normalized count data after being both centered and scaled (Zscore).

### See Also

The function calls the function `ComplexHeatmap::Heatmap()` in order to plot the Heatmaps.

### Examples

```
## data importation
data("RawCounts_Antoszewski2022_MOUSEsub500")
## No time points. We take only two groups for the speed of the example
dataT1wt <- RawCounts_Antoszewski2022_MOUSEsub500[seq_len(200), seq_len(7)]

## Preprocessing with Results of DEanalysisGlobal()
resDATAprepSE <- DATAprepSE(RawCounts=dataT1wt,
                             Column.gene=1,
                             Group.position=1,
                             Time.position=NULL,
                             Individual.position=2)

##-----##
## DE analysis
resDET1wt <- DEanalysisGlobal(SEres=resDATAprepSE,
                             pval.min=0.05,
                             pval.vect.t=NULL,
                             log.FC.min=1,
                             LRT.suppl.info=FALSE,
                             Plot.DE.graph=FALSE,
                             path.result=NULL,
                             Name.folder.DE=NULL)

##-----##
resHeatmap <- DEplotHeatmaps(SEresDE=resDET1wt,
                             ColumnsCriteria=3, ## Specific genes N1haT1ko
                             Set.Operation="union",
                             NbGene.analysis=20,
                             Color.Group=NULL,
                             SizeLabelRows=5,
                             SizeLabelCols=5,
                             Display.plots=TRUE,
                             Save.plots=FALSE)
```

---

DEplotVennBarplotGroup

*Venn barplot of DE genes across pairs of biological conditions.*

---

**Description**

The function takes as input a binary matrix or data.frame with  $N_g$  rows and  $((N_{bc} - 1) \times N_{bc})/2$  columns with  $N_g$  the number of genes and  $N_{bc}$  the number of biological conditions. The number of 1 in the  $n$ -th row gives the number of pairs of biological conditions where the gene  $n$  is DE. We consider that a set of pairs of biological conditions forms an intersection if there is at least one gene which is DE for each of these pairs of biological conditions, but not for the others.

The function calls the `UpSetR::upset()` function in order to plot the number of genes for each possible intersection in an UpSet plot (Venn diagram displayed as a barplot).

**Usage**

```
DEplotVennBarplotGroup(Mat.DE.pair.group)
```

**Arguments**

Mat.DE.pair.group

Binary matrix or data.frame with  $N_g$  rows and  $((N_{bc} - 1) * N_{bc})/2$  columns with  $N_{bc}$  the number of biological conditions.

**Value**

The function plots the number of genes for each possible intersection in a UpSet plot.

**See Also**

The function

- calls the function `UpSetR::upset()` in order to plot the UpSet plot.
- is called by the functions `DEanalysisGroup()` and `DEanalysisTimeAndGroup()`.

**Examples**

```
set.seed(1994)
##-----##
## Binary matrix
Bin.Table.G <- matrix(c(sample(c(0,1), replace=TRUE, size=240,c(0.75,0.35)),
                        sample(c(0,1), replace=TRUE, size=240,c(0.3,0.7)),
                        rep(0,18)),
                    ncol=6, byrow=TRUE)
colnames(Bin.Table.G) <- c(".A..B.", ".A..C.", ".A..D.",
                          ".B..C.", ".B..D.", ".C..D.")
##-----##
## Results
res.t.upset <- DEplotVennBarplotGroup(Mat.DE.pair.group=Bin.Table.G)
print(res.t.upset$Upset.global)
print(res.t.upset$Upset.threshold)
```

---

DEplotVennBarplotTime *Venn barplot of DE genes across time.*

---

## Description

The function takes as input two matrix or data.frame

- a binary matrix or data.frame with  $N_g$  rows corresponding to genes and  $T - 1$  columns corresponding to times (with  $T$  the number of time points). A '1' in the n-th row and i-th column means that the n-th gene is differentially expressed (DE) at time  $t_i$ , compared with the reference time  $t_0$ .
- a numeric matrix or data.frame with  $N_g$  rows corresponding to genes and  $T - 1$  columns corresponding to times. The element in n-th row and i-th column corresponds to the  $\log_2$  fold change between the time  $t_i$  and the reference time  $t_0$  for the n-th gene. If the gene is DE and the sign is positive, then the gene n will be considered as over-expressed (up-regulated) at time  $t_i$ . If the gene is DE and the sign is negative, then the gene n will be considered as under-expressed (down-regulated) at time  $t_i$ .

## Usage

```
DEplotVennBarplotTime(table.DE.time, Log2.FC.matrix)
```

## Arguments

`table.DE.time` Binary matrix or data.frame (table filled with 0 and 1) with  $N_g$  rows and  $T - 1$  columns with  $N_g$  the number of genes and  $T$  the number of time points.

`Log2.FC.matrix` Numeric matrix or data.frame with  $N_g$  rows and  $T - 1$  columns.

## Value

The function plots

- the number of genes per time patterns in an UpSet plot (Venn diagram displayed as a barplot) with the R function `UpSetR::upset()`. By temporal pattern, we mean the set of times  $t_i$  such that the gene is DE between  $t_i$  and the reference time  $t_0$ .
- a similar UpSet plot where each bar is split in different colors corresponding to all possible numbers of DE times where genes are over expressed in a given temporal pattern.

## See Also

The function

- calls the function `UpSetR::upset()` in order to plot the UpSet plot.
- is called by the functions `DEanalysisTime()` and `DEanalysisTimeAndGroup()`.

**Examples**

```

set.seed(1994)
Nb.Time <- 4 ## Number of time measurement
##-----##
table.DE.time.ex <- matrix(sample(c(0,1), replace=TRUE,
                                size=40*(Nb.Time-1), c(0.2, 0.8)),
                           ncol=Nb.Time-1)
colnames(table.DE.time.ex) <- paste0("t", 1:(Nb.Time-1))
##-----##
Log2FC.mat.ex <- matrix(round(rnorm(n=40*(Nb.Time-1), mean=0, sd=1),
                               digits=2),
                        ncol=(Nb.Time-1))
colnames(Log2FC.mat.ex) <- paste0("t", 1:(Nb.Time-1))
##-----##
res.test.VennBarplot <- DEplotVennBarplotTime(table.DE.time=table.DE.time.ex,
                                             Log2.FC.matrix=Log2FC.mat.ex)

print(res.test.VennBarplot$Upset.graph)
print(res.test.VennBarplot$Upset.graph.with.nb.over)
res.test.VennBarplot$DE.pattern.t.01.sum

```

DEplotVolcanoMA

*Volcano and MA graphs***Description**

The function returns Volcano plots and MA plots from the results of our function [DEanalysisGlobal\(\)](#).

**Usage**

```

DEplotVolcanoMA(
  SEresDE,
  NbGene.plotted = 2,
  SizeLabel = 3,
  Display.plots = FALSE,
  Save.plots = FALSE
)

```

**Arguments**

SEresDE	A SummarizedExperiment class object. Output from <a href="#">DEanalysisGlobal()</a> (see Examples).
NbGene.plotted	Non negative integer. The algorithm computes the sum of all the absolute $\log_2$ fold change present in the element DE.results of Res.DE.analysis for each gene. Only the highest NbGene.plotted genes are plotted in the volcano and MA plots. By default, NbGene.plotted=2.
SizeLabel	Numeric. Give the size of the names of plotted genes. By default, SizeLabel=3.
Display.plots	TRUE or FALSE. FALSE as default. If TRUE, all graphs will be plotted. Otherwise no graph will be plotted.

`Save.plots` TRUE or FALSE or a Character. FALSE as default. Path to save the Volcano and MA plots. If NULL, the Volcano and MA plots will not be saved in a sub folder in `path.result`.

If `path.result` contains a sub folder entitled "VolcanoPlots", all the Volcano plots will be saved in the sub folder "VolcanoPlots". Otherwise, a sub folder entitled "VolcanoPlots" will be created in `path.result` and all the Volcano plots will be saved in the sub folder created.

If `path.result` contains a sub folder entitled "MAplots", all the MA plots will be saved in the sub folder "MAplots". Otherwise, a sub folder entitled "MAplots" will be created in `path.result` and all the MA plots will be saved in the sub folder created.

### Details

- If data belong to different time points only, the function returns  $T - 1$  volcano and MA plots (with  $T$  the number of time measurements), corresponding to the  $\log_2$  fold change between each time  $t_i$  and the reference time  $t_0$ , for all  $i > 0$ .
- If data belong to different biological conditions only, the function returns  $(N_{bc} * (N_{bc} - 1))/2$  volcano and MA plots (with  $N_{bc}$  the number of biological conditions), corresponding to the  $\log_2$  fold change between each pair of biological condition.
- If data belong to different biological conditions and time points, the function returns
  - $(T - 1) * N_{bc}$  volcano and MA plots, corresponding to the  $\log_2$  fold change between each time  $t_i$  and the reference time  $t_0$ , for all biological condition.
  - $((T - 1) * N_{bc} * (N_{bc} - 1))/2$  volcano and MA plots, corresponding to the  $\log_2$  fold change between each pair of biological conditions, for all fixed time point.

### Value

The function returns the same SummarizedExperiment class object `SEresDE` with Volcano plots and MA plots from the results of our function `DEanalysisGlobal()`, all saved in the metadata `Results[[2]][[3]]` of `SEresDE`.

### See Also

The function calls the output of `DEanalysisGlobal()`.

### Examples

```
## data importation
data(RawCounts_Antoszewski2022_MOUSEsub500)
## No time points. We take only two groups for the speed of the example
dataT1wt <- RawCounts_Antoszewski2022_MOUSEsub500[seq_len(200), seq_len(7)]

## Preprocessing with Results of DEanalysisGlobal()
resDATAprepSE <- DATAprepSE(RawCounts=dataT1wt,
                             Column.gene=1,
                             Group.position=1,
                             Time.position=NULL,
                             Individual.position=2)
```

```
##-----##
## DE analysis
resDET1wt <- DEanalysisGlobal(SEres=resDATAprepSE,
                             pval.min=0.05,
                             pval.vect.t=NULL,
                             log.FC.min=1,
                             LRT.supp.info=FALSE,
                             Plot.DE.graph=FALSE,
                             path.result=NULL,
                             Name.folder.DE=NULL)

##-----##
## Volcano MA
resVolcanoMA <- DEplotVolcanoMA(SEresDE=resDET1wt,
                                 NbGene.plotted=5,
                                 Display.plots=TRUE,
                                 Save.plots=FALSE)
```

---

DEresultGroup

*Intermediate analysis when samples belong to different biological conditions*


---

## Description

This function realizes the intermediary steps of the analysis of the function [DEanalysisGroup\(\)](#).

## Usage

```
DEresultGroup(
  DESeq.result,
  LRT.supp.info = TRUE,
  pval.min = 0.05,
  log.FC.min = 1
)
```

## Arguments

DESeq.result	Output from the function <a href="#">DESeq2::DESeq()</a> .
LRT.supp.info	TRUE or FALSE. If TRUE, the algorithm realizes another statistical test in order to detect if, among all biological conditions and/or times, at least one has a different behavior than the others (see the input test in <a href="#">DESeq2::DESeq()</a> ).
pval.min	Numeric value between 0 and 1. A gene will be considered as differentially expressed (DE) between two biological conditions if its Benjamini-Hochberg adjusted p-value (see <a href="#">stats::p.adjust()</a> ) is below the threshold pval.min. Default value is 0.05.
log.FC.min	Non negative numeric value. If the log <sub>2</sub> fold change between biological conditions or times has an absolute value below the threshold log.FC.min, then the gene is not selected even if is considered as DE. Default value is 1. If log.FC.min=0, all DE genes will be kept.



```

## Preprocessing step
resDATAprepSEmus1 <- DATAprepSE(RawCounts=RawCounts_T1Wt,
                                Column.gene=1,
                                Group.position=1,
                                Time.position=NULL,
                                Individual.position=2)

DESeq2preprocess <- S4Vectors::metadata(resDATAprepSEmus1)$DESeq2obj
DESeq2obj <- DESeq2preprocess$DESeq2preprocessing

##-----##
dds.DE.G <- DESeq2::DESeq(DESeq2obj)

res.sum.G <- DEresultGroup(DESeq.result=dds.DE.G,
                           LRT.suppl.info=FALSE,
                           log.FC.min=1,
                           pval.min=0.05)

```

---

DEresultGroupPerTime *Intermediate analysis when samples belong to different biological conditions and different time points.*

---

## Description

This function realizes the intermediate steps of the analysis of the function [DEanalysisTimeAndGroup\(\)](#).

## Usage

```

DEresultGroupPerTime(
  DESeq.result,
  LRT.suppl.info = TRUE,
  pval.min = 0.05,
  log.FC.min = 1
)

```

## Arguments

DESeq.result	Output from the function <a href="#">DESeq2::DESeq()</a> .
LRT.suppl.info	TRUE or FALSE. If TRUE, the algorithm realizes another statistical test in order to detect if, among all biological conditions and/or times, at least one has a different behavior than the others (see the input test in <a href="#">DESeq2::DESeq()</a> ).
pval.min	Numeric value between 0 and 1. A gene will be considered as differentially expressed (DE) between two biological conditions if its Benjamini-Hochberg adjusted p-value (see <a href="#">stats::p.adjust()</a> ) is below the threshold pval.min. Default value is 0.05.



`log.FC.min` Non negative numeric value. If the  $\log_2$  fold change between biological conditions or times has an absolute value below the threshold `log.FC.min`, then the gene is not selected even if is considered as DE. Default value is 1. If `log.FC.min=0`, all DE genes will be kept.

## Value

The function returns the same `DESeqDataSet` class object `DESeq.result` with the following results, saved in the metadata `DEresultsTimeGroup` of `DESeq.result`:

- a data.frame (output `DEsummary` of `DEresultsTimeGroup`) which contains
  - pvalues,  $\log_2$  fold change and DE genes between each pairs of biological conditions for a fixed time  $t_i$  (except the reference time  $t_0$ ).
  - DE specific genes per biological condition for a fixed time  $t_i$  (except the reference time  $t_0$ ).
- inputs for the functions : [DEplotBarplot\(\)](#), [DEplotBarplotTime\(\)](#), [DEplotVennBarplotGroup\(\)](#), [DEplotVennBarplotTime\(\)](#), [DEplotBarplotFacetGrid\(\)](#), [DEplotAlluvial\(\)](#).

## See Also

The output of the function are used by the main function [DEanalysisTimeAndGroup\(\)](#).

## Examples

```
data("RawCounts_Schleiss2021_CLLsub500")
## We take only the first three times (both group) for the speed of
## the example
Index3t<-c(2:4,11:13,20:22, 29:31,38:40,47:49)
RawCounts_3t<-RawCounts_Schleiss2021_CLLsub500[seq_len(200), c(1,Index3t)]

## Preprocessing step
resDATAprepSEleuk <- DATAprepSE(RawCounts=RawCounts_3t,
                                 Column.gene=1,
                                 Group.position=2,
                                 Time.position=4,
                                 Individual.position=3)

DESeq2preprocess <- S4Vectors::metadata(resDATAprepSEleuk)$DESeq2obj
DESeq2obj <- DESeq2preprocess$DESeq2preprocessing

##-----##
dds.DE <- DESeq2::DESeq(DESeq2obj)
##
res.G.T.2 <- DEresultGroupPerTime(DESeq.result=dds.DE,
                                 LRT.suppl.info=FALSE,
                                 log.FC.min=1,
                                 pval.min=0.05)
```

---

GSEAPreprocessing      *GSEA preprocessing for official software and online tools.*

---

### Description

The function returns, from the output of `DEanalysisGlobal()`, specific files designed to be used as input for several online tools and software given in the section Value.

### Usage

```
GSEAPreprocessing(
  SEresDE,
  ColumnsCriteria,
  Set.Operation,
  Rnk.files = TRUE,
  Save.files = FALSE
)
```

### Arguments

<code>SEresDE</code>	A SummarizedExperiment class object. Output from <code>DEanalysisGlobal()</code> (see Examples).
<code>ColumnsCriteria</code>	A vector of integers where each integer indicates a column of <code>SummarizedExperiment::rowData(SEresDE)</code> . These columns should either contain only binary values, or may contain other numerical value, in which case extracted outputs from <code>SEresDE</code> will be those with $>0$ values (see Details).
<code>Set.Operation</code>	A character. The user must choose between "union" (default), "intersect", "set-diff" (see Details).
<code>Rnk.files</code>	TRUE or FALSE. TRUE as default. If TRUE, the <code>rnk</code> files generated by the function (used by the GSEA software) will be saved if <code>Save.files=TRUE</code> and <code>path.result</code> of <code>DEanalysisGlobal()</code> is not NULL. Otherwise the <code>rnk</code> files will not be generated.
<code>Save.files</code>	TRUE or FALSE or a Character. If <code>Save.files=TRUE</code> and the <code>path.result</code> of <code>DEanalysisGlobal()</code> is not NULL, all files will be saved in "2_Supervised-Analysis_Name.folder.DE/ 2-5_Enrichment_analysis_Name.folder.DE/ 2-5-2_EnrichmentGO_software_preprocessing". If <code>Save.files</code> is a character, it must be a path and all files will be saved in the sub-folder "EnrichmentGO_software_preprocessing". Otherwise, the different files will not be saved.

### Details

We have the following three cases:

- If `Set.Operation="union"` then the rows extracted from the different datasets (raw counts, normalized data and `SummarizedExperiment::rowData(SEresDE)`) included in the SummarizedExperiment class object `SEresDE` are those such that the sum of the selected columns of

SummarizedExperiment::rowData(SeresDE) given in ColumnsCriteria is >0. This means that the selected genes are those having at least one '1' in one of the selected columns.

- If Set.Operation="intersect" then the rows extracted from the different datasets (raw counts, normalized data and SummarizedExperiment::rowData(SeresDE)) included in the SummarizedExperiment class object SeresDE are those such that the product of the selected columns of SummarizedExperiment::rowData(SeresDE) given in ColumnsCriteria is >0. This means that the selected genes are those having '1' in all of the selected columns.
- If Set.Operation="setdiff" then the rows extracted from the different datasets (raw counts, normalized data and SummarizedExperiment::rowData(SeresDE)) included in the SummarizedExperiment class object SeresDE are those such that only one element of the selected columns of SummarizedExperiment::rowData(SeresDE) given in ColumnsCriteria is >0. This means that the selected genes are those having '1' in only one of the selected columns.

## Value

The function returns

- A vector of character containing gene names specified by ColumnsCriteria and Set.Operation.
- A vector of character containing all gene names
- And, in case where Save.files=TRUE and the path.result of DEanalysisGlobal() is not NULL, specific files designed to be used as input for the following online tools and software :
  - GSEA : <https://www.gsea-msigdb.org/gsea/index.jsp>
  - DAVID : <https://david.ncifcrf.gov/tools.jsp>
  - WebGestalt : <http://www.webgestalt.org>
  - gProfiler : <https://biit.cs.ut.ee/gprofiler/gost>
  - Panther : <http://www.pantherdb.org>
  - ShinyGO : <http://bioinformatics.sdstate.edu/go/>
  - Enrichr : <https://maayanlab.cloud/Enrichr/>
  - GOrilla : <http://cbl-gorilla.cs.technion.ac.il>.

## Examples

```
data(RawCounts_Antoszewski2022_MOUSEsub500)
## No time points. We take only two groups for the speed of the example
RawCounts_T1Wt <- RawCounts_Antoszewski2022_MOUSEsub500[, seq_len(7)]
##-----##
## Preprocessing
resDATAprepSE <- DATAprepSE(RawCounts=RawCounts_T1Wt,
                             Column.gene=1,
                             Group.position=1,
                             Time.position=NULL,
                             Individual.position=2)
##-----##
## DE analysis
resDET1wt <- DEanalysisGlobal(Seres=resDATAprepSE,
                              pval.min=0.05,
                              pval.vect.t=NULL,
                              log.FC.min=1,
```

```

LRT.suppl.info=FALSE,
Plot.DE.graph=TRUE,
path.result=NULL,
Name.folder.DE=NULL)

##-----##
resGp <- GSEApresprocessing(SEResDE=resDET1wt,
                           ColumnsCriteria=2,
                           Set.Operation="union",
                           Rnk.files=TRUE,
                           Save.files=FALSE)

```

---

GSEAQuickAnalysis      *GSEA analysis with gprofiler2*

---

## Description

The function realizes, from the outputs of [DEanalysisGlobal\(\)](#), an enrichment analysis (GSEA) of a subset of genes with the R package `gprofiler2`.

## Usage

```

GSEAQuickAnalysis(
  Internet.Connection = FALSE,
  SEResDE,
  ColumnsCriteria = 1,
  ColumnsLog2ordered = NULL,
  Set.Operation = "union",
  Organism = "hsapiens",
  MaxNumberGO = 20,
  Background = FALSE,
  Display.plots = TRUE,
  Save.plots = FALSE
)

```

## Arguments

`Internet.Connection`

TRUE or FALSE. FALSE by default. If the user is sure to have an internet connection, the user must set `Internet.Connection=TRUE`, otherwise, the algorithm will not run.

`SEResDE`

A `SummarizedExperiment` class object. Output from [DEanalysisGlobal\(\)](#) (see Examples).

`ColumnsCriteria`

A vector of integers where each integer indicates a column of `SummarizedExperiment::rowData(SEResDE)`. These columns should either contain only binary values, or may contain other numerical value, in which case extracted outputs from `SEResDE` will be those with  $>0$  values (see Details).

ColumnsLog2ordered	NULL or a vector of integers. If ColumnsLog2ordered is a vector of integers, it corresponds to the columns number of Res.DE.analysis\$DE.results, the output of <a href="#">DEanalysisGlobal()</a> , which must contains $\log_2$ fold change values (see Details).
Set.Operation	A character. The user must choose between "union" (default), "intersect", "set-diff" (see Details).
Organism	A character indicating the organism where data were taken from. See vignette of the R package <a href="#">gprofiler2</a> for supported organisms. See <a href="#">gprofiler2::gost()</a> .
MaxNumberGO	An integer. The user can select the MaxNumberGO most important Gene Ontology (GO) names to be plotted in a lollipop graph. By default, MaxNumberGO=20.
Background	TRUE or FALSE. If TRUE, the statistical enrichment analysis to find over-representation of functions from Gene Ontology (GO) and biological pathways (e.g. KEGG) will be done by comparing the functions and biological pathways among the selected DE genes with those associated with all genes in Res.DE.analysis\$DE.results. If FALSE, the statistical enrichment analysis will be done by comparing the functions and biological pathways among the selected DE genes with all functions and biological pathways included in the database of <a href="#">gprofiler2</a> (link in See Also). See also <a href="#">gprofiler2::gost()</a> .
Display.plots	TRUE or FALSE. TRUE as default. If TRUE, all graphs will be plotted. Otherwise no graph will be plotted.
Save.plots	TRUE or FALSE or a Character. If Save.plots=TRUE and the output path.result of <a href="#">DEanalysisGlobal()</a> is not NULL, all files will be saved in "2_Supervised-Analysis_Name.folder.DE/ 2-5_Enrichment_analysis_Name.folder.DE/ 2-5-1_gprofiler2_results_Name.folder.DE", with Name.folder.DE an input of <a href="#">DEanalysisGlobal()</a> . If Save.plots is a character, it must be a path and all files will be saved in the sub-folder "gprofiler2_results_Name.folder.DE". Otherwise, the different files will not be saved.

## Details

If ColumnsLog2ordered is a vector of integers, the rows of Res.DE.analysis\$DE.results (corresponding to genes) will be decreasingly ordered according to the sum of absolute  $\log_2$  fold change (the selected columns must contain  $\log_2$  fold change values) before the enrichment analysis. The enrichment analysis will take into account the genes order as the first genes will be considered to have the highest biological importance and the last genes the lowest. See the input ordered\_query of [gprofiler2::gost\(\)](#) and the vignette of [gprofiler2](#) for more details.

We have the following three cases:

- If Set.Operation="union" then the rows extracted from the different datasets (raw counts, normalized data and SummarizedExperiment::rowData(SEResDE)) included in the SummarizedExperiment class object SEResDE are those such that the sum of the selected columns of SummarizedExperiment::rowData(SEResDE) given in ColumnsCriteria is >0. This means that the selected genes are those having at least one '1' in one of the selected columns.
- If Set.Operation="intersect" then the rows extracted from the different datasets (raw counts, normalized data and SummarizedExperiment::rowData(SEResDE)) included in the SummarizedExperiment class object SEResDE are those such that the product of the selected

columns of `SummarizedExperiment::rowData(SeresDE)` given in `ColumnsCriteria` is  $>0$ . This means that the selected genes are those having '1' in all of the selected columns.

- If `Set.Operation="setdiff"` then the rows extracted from the different datasets (raw counts, normalized data and `SummarizedExperiment::rowData(SeresDE)`) included in the `SummarizedExperiment` class object `SeresDE` are those such that only one element of the selected columns of `SummarizedExperiment::rowData(SeresDE)` given in `ColumnsCriteria` is  $>0$ . This means that the selected genes are those having '1' in only one of the selected columns.

## Value

The function returns the same `SummarizedExperiment` class object `SeresDE` with

- a data.frame which contains the outputs of `gprofiler2::gost()`
- a Manhattan plot showing all GO names according to their pvalue
- a lollipop graph showing the `MaxNumberGO` most important GO.

saved in the metadata `Results[[2]][[5]]` of `SeresDE`.

The Manhattan plot and the lollipop graph are plotted if `Display.plots=TRUE`.

## See Also

The function uses the R package `gprofiler2` <https://cran.r-project.org/web/packages/gprofiler2/vignettes/gprofiler2.html>.

The R package `gprofiler2` provides an R interface to the web toolset `g:Profiler` <https://biit.cs.ut.ee/gprofiler/gost>.

## Examples

```
## data importation
data(RawCounts_Antoszewski2022_MOUSEsub500)
## No time points. We take only two groups for the speed of the example
dataT1wt <- RawCounts_Antoszewski2022_MOUSEsub500[seq_len(200), seq_len(7)]

## Preprocessing with Results of DEanalysisGlobal()
resDATAprepSE <- DATAprepSE(RawCounts=dataT1wt,
                             Column.gene=1,
                             Group.position=1,
                             Time.position=NULL,
                             Individual.position=2)

##-----##
## Internet is needed in order to run the following lines of code because
## gprofiler2 needs an internet connection
## DE analysis
# resDET1wt <- DEanalysisGlobal(Seres=resDATAprepSE,
#                               # pval.min=0.05,
#                               # pval.vect.t=NULL,
#                               # log.FC.min=1,
#                               # LRT.supp.info=FALSE,
#                               # Plot.DE.graph=FALSE,
```

```

#           path.result=NULL,
#           Name.folder.DE=NULL)
#####
# resGs <- GSEAQuickAnalysis(Internet.Connection=TRUE,
#                             SEresDE=resDET1wt,
#                             ColumnsCriteria=3,
#                             ColumnsLog2ordered=NULL,
#                             Set.Operation="union",
#                             Organism="mmusculus",
#                             MaxNumberGO=20,
#                             Background=FALSE,
#                             Display.plots=TRUE,
#                             Save.plots=FALSE)

```

---

HCPCanalysis

*Hierarchical clustering analysis with HCPC (Main function)*


---

## Description

The functions performs a hierarchical clustering on results from a factor analysis with the R function [FactoMineR::HCPC\(\)](#).

## Usage

```

HCPCanalysis(
  SEresNorm,
  DATAnorm = TRUE,
  gene.deletion = NULL,
  sample.deletion = NULL,
  Plot.HCPC = FALSE,
  Color.Group = NULL,
  Phi = 25,
  Theta = 140,
  epsilon = 0.2,
  Cex.point = 0.7,
  Cex.label = 0.7,
  motion3D = FALSE,
  path.result = NULL,
  Name.folder.hcpc = NULL
)

```

## Arguments

SEresNorm	Results of the function <a href="#">DATAnormalization()</a> .
DATAnorm	TRUE or FALSE. TRUE as default. TRUE means the function uses the normalized data. FALSE means the function uses the raw counts data.

<code>gene.deletion</code>	NULL or a vector of characters or a vector of integers. NULL as default. If <code>gene.deletion</code> is a vector of characters, all genes with names in <code>gene.deletion</code> will be deleted from the data set as input <code>RawCounts</code> of our function <code>DATAprepSE()</code> . If <code>gene.deletion</code> is a vector of integers, all the corresponding row numbers will be deleted from the data set as input <code>RawCounts</code> of our function <code>DATAprepSE()</code> . If <code>gene.deletion=NULL</code> all genes will be used in the construction of the PCA.
<code>sample.deletion</code>	NULL or a vector of characters or a vector of integers. NULL as default. If <code>sample.deletion</code> is a vector of characters, all samples with names in <code>sample.deletion</code> will not be used in the construction of the PCA. If <code>sample.deletion</code> is a vector of integers, all the corresponding column numbers will not be used in the construction of the PCA from the data set as input <code>RawCounts</code> of our function <code>DATAprepSE()</code> . If <code>sample.deletion=NULL</code> all samples will be used in the construction of the PCA.
<code>Plot.HCPC</code>	TRUE or FALSE. FALSE as default. If TRUE, all graphs will be plotted. Otherwise no graph will be plotted.
<code>Color.Group</code>	NULL or a data.frame with $N_{bc}$ rows and two columns where $N_{bc}$ is the number of biological conditions. If <code>Color.Group</code> is a data.frame, the first column must contain the name of each biological condition and the second column must contain the colors associated to each biological condition. If <code>Color.Group=NULL</code> , the function will automatically attribute a color for each biological condition. If samples belong to different time points only, <code>Color.Group</code> will not be used.
<code>Phi</code>	Angle defining the colatitude direction for the 3D PCA plot (see Details in <code>graphics::persp()</code> ).
<code>Theta</code>	Angle defining the azimuthal direction for the 3D PCA plot (see Details in <code>graphics::persp()</code> ).
<code>epsilon</code>	Non negative numeric value giving the length between points and their labels in all PCA plots which are not automatically plotted by <code>FactoMineR::PCA()</code> .
<code>Cex.point</code>	Non negative numeric value giving the size of points in all PCA plots which are not automatically plotted by <code>FactoMineR::PCA()</code> .
<code>Cex.label</code>	Non negative numeric value giving the size of the labels associated to each point of the all PCA graphs which are not automatically plotted by <code>FactoMineR::PCA()</code> .
<code>motion3D</code>	TRUE or FALSE. If TRUE, the 3D PCA plots will also be plotted in a rgl window (see <code>plot3Drgl::plotrgl()</code> ) allowing to interactively rotate and zoom.
<code>path.result</code>	Character or NULL. Path to save all results. If <code>path.result</code> contains a sub folder entitled "1_UnsupervisedAnalysis_Name.folder.hcpc" and a sub sub folder, "1-3_HCPCanalysis_Name.folder.hcpc" all results will be saved in the sub folder "1_UnsupervisedAnalysis_Name.folder.hcpc/1-3_HCPCanalysis_Name.folder.hcpc". Otherwise, a sub folder entitled "1_UnsupervisedAnalysis_Name.folder.hcpc" and/or a sub sub folder "1-3_HCPCanalysis_Name.folder.hcpc" will be created in <code>path.result</code> and all results will be saved in "1_UnsupervisedAnalysis_Name.folder.hcpc/1-3_HCPCanalysis_Name.folder.hcpc". If NULL, the results will not be saved in a folder. NULL as default.
<code>Name.folder.hcpc</code>	Character or NULL. If <code>Name.folder.hcpc</code> is a character, the folder and sub folder names which will contain the PCA graphs will respectively be "1_UnsupervisedAnalysis_Name.folder.hcpc" and "1-3_HCPCanalysis_Name.folder.hcpc".



Otherwise, the folder and sub folder names will respectively be "1\_Unsuper-visedAnalysis" and "1-3\_HCPCanalysis".

### Details

All results are built from the results of our function `DATAnormalization()`.

The number of clusters is automatically selected by `FactoMineR::HCPC()` and is described in the section Details of `FactoMineR::HCPC()`.

### Value

The function returns the same SummarizedExperiment class object `SEresNorm` with the outputs from the function `FactoMineR::HCPC()`, (saved in the metadata `Results[[1]][[3]]` of `SEresNorm`)

- a dendrogram (also called hierarchical tree) using the function `factoextra::fviz_dend()`
- one 2D PCA and two 3D PCA produced by the function `PCAgraphics()` where samples are colored with different colors for different clusters. The two 3D PCA graphs are identical but one of them will be opened in a `rgl` window (see `plot3Drgl::plotrgl()`) allowing to interactively rotate and zoom. The interactive 3D graph will be plotted only if `motion3D=TRUE`.
- A graph indicating for each sample, its cluster and the time and/or biological condition associated to the sample.
- the outputs of `FactoMineR::HCPC()`.

### See Also

The function calls the functions `PCArealization()` and `FactoMineR::HCPC()`. The function `FactoMineR::HCPC()` will take as input the output of `PCArealization()`.

### Examples

```
## Simulation raw counts
resSIMcount <- RawCountsSimulation(Nb.Group=2, Nb.Time=3, Nb.per.GT=4,
                                   Nb.Gene=10)

## Preprocessing step
resDATAprepSE <- DATAprepSE(RawCounts=resSIMcount$Sim.dat,
                             Column.gene=1,
                             Group.position=1,
                             Time.position=2,
                             Individual.position=3)

## Normalization
resNorm <- DATAnormalization(SEres=resDATAprepSE,
                             Normalization="rle",
                             Plot.Boxplot=FALSE,
                             Colored.By.Factors=FALSE)

##-----##
reshCPCanalysis <- HCPCanalysis(SEresNorm=resNorm,
                                DATAnorm=TRUE,
                                sample.deletion=NULL,
                                gene.deletion=NULL,
                                Plot.HCPC=TRUE,
```

```

Color.Group=NULL,
Phi=25, Theta=140,
Cex.point=1, Cex.label=0.6, epsilon=0.4,
motion3D=FALSE,
path.result=NULL,
Name.folder.hcpc=NULL)

```

---

MFUZZanalysis

*Clustering of temporal patterns (Main function).*


---

### Description

The function performs a soft clustering of temporal patterns based on the fuzzy c-means algorithm using the R package Mfuzz.

### Usage

```

MFUZZanalysis(
  SEresNorm,
  DATAnorm = TRUE,
  DataNumberCluster = NULL,
  Method = "hcpc",
  Max.clust = 6,
  Membership = 0.5,
  Min.std = 0.1,
  Plot.Mfuzz = TRUE,
  path.result = NULL,
  Name.folder.mfuzz = NULL
)

```

### Arguments

SEresNorm	Results of the function <a href="#">DATAnormalization()</a> .
DATAnorm	TRUE or FALSE. TRUE as default. TRUE means the function uses the normalized data. FALSE means the function uses the raw counts data.
DataNumberCluster	Data.frame or NULL. NULL as default. If DataNumberCluster is a data.frame where the first column contains the name of the biological conditions and the second the number of cluster selected for each biological condition. If DataNumberCluster=NULL, a number of clusters will be automatically computed for each biological condition (see <a href="#">MFUZZclustersNumber()</a> ).
Method	"kmeans" or "hcpc". The method used for selecting the number of cluster to be used for the temporal cluster analysis (see <a href="#">Details</a> ). Only used if DataNumberCluster is not NULL.
Max.clust	Integer strictly superior to 1 indicating the maximum number of clusters. Max.clust will be used only if DataNumberCluster=NULL

Membership	Numeric value between 0 and 1. For each cluster, genes with membership values below the threshold Membership will not be displayed. The membership values correspond to the probability of gene to belong to each cluster.
Min.std	Numeric positive value. All genes where their standard deviations are smaller than the threshold Min.std will be excluded.
Plot.Mfuzz	TRUE or FALSE. TRUE as default. If TRUE, all graphs will be plotted. Otherwise no graph will be plotted.
path.result	Character or NULL. Path to save all results. If path.result contains a sub folder entitled "1_UnsupervisedAnalysis_Name.folder.mfuzz" and a sub sub folder, "1-4_MFUZZanalysis_Name.folder.mfuzz" all results will be saved in the sub folder "1_UnsupervisedAnalysis_Name.folder.mfuzz/1-4_MFUZZanalysis_Name.folder.mfuzz". Otherwise, a sub folder entitled "1_UnsupervisedAnalysis_Name.folder.mfuzz" and/or a sub sub folder "1-4_MFUZZanalysis_Name.folder.mfuzz" will be created in path.result and all results will be saved in "1_UnsupervisedAnalysis_Name.folder.mfuzz/1-4_MFUZZanalysis_Name.folder.mfuzz". If NULL, the results will not be saved in a folder. NULL as default.
Name.folder.mfuzz	Character or NULL. If Name.folder.mfuzz is a character, the folder and sub folder names which will contain the PCA graphs will respectively be "1_UnsupervisedAnalysis_Name.folder.mfuzz" and "1-4_MFUZZanalysis_Name.folder.mfuzz". Otherwise, the folder and sub folder names will respectively be "1_UnsupervisedAnalysis" and "1-4_MFUZZanalysis".

## Details

All results are built from the results of our function `DATanormalization()`.

The Mfuzz package works with datasets where rows correspond to genes and columns correspond to times. If `RawCounts` (input of our function `DATaprepSE()`) contains several replicates per time, the algorithm computes the mean of replicates for each gene before using `Mfuzz::mfuzz()`. When there are several biological conditions, the algorithm realizes the `Mfuzz::mfuzz()` analysis for each biological condition.

## Value

The function returns the same `SummarizedExperiment` class object `SEresNorm` with the different elements below (saved in the metadata `Results[[1]][[4]]` of `SEresNorm`)

- the final data used for the Mfuzz analysis (see Details).
- the cluster associated to each gene.
- plots generated by `MFUZZclustersNumber()` and `Mfuzz::mfuzz.plot2()` for each biological condition.

## See Also

The function uses the function `MFUZZclustersNumber()` to compute the optimal number of cluster for each biological condition with the kmeans method.

**Examples**

```

## Data simulation
set.seed(33)
DATAclustSIM <- matrix(rnorm(12*10*3, sd=0.2,
                           mean=rep(c(rep(c(1, 6, 9, 4, 3, 1,
                                           6.5, 0.7, 10), times=2),
                                       rep(c(2, 3.6, 3.7, 5, 7.9, 8,
                                           7.5, 3.5, 3.4), times=2))),
                           each=10)),
                      nrow=30, ncol=12)
DATAclustSIM <- floor(DATAclustSIM*100)
##
colnames(DATAclustSIM) <- c("G1_t0_r1", "G1_t1_r1", "G1_t2_r1",
                           "G1_t0_r2", "G1_t1_r2", "G1_t2_r2",
                           "G2_t0_r3", "G2_t1_r3", "G2_t2_r3",
                           "G2_t0_r4", "G2_t1_r4", "G2_t2_r4")
##-----##
## Plot the temporal expression of each individual
graphics::matplot(t(rbind(DATAclustSIM[, 1:3], DATAclustSIM[, 4:6],
                           DATAclustSIM[, 7:9], DATAclustSIM[, 10:12])),
                  col=rep(c("black", "red"), each=6*10),
                  xlab="Time", ylab="Gene expression", type=c("b"), pch=19)
##-----##
## Preprocessing step
DATAclustSIM <- data.frame(DATAclustSIM)

resDATAprepSE <- DATAprepSE(RawCounts=DATAclustSIM,
                             Column.gene=NULL,
                             Group.position=1,
                             Time.position=2,
                             Individual.position=3)

## Normalization
resNorm <- DATAnormalization(SEres=resDATAprepSE,
                             Normalization="rle",
                             Plot.Boxplot=FALSE,
                             Colored.By.Factors=FALSE)
##-----##
resMFUZZ <- MFUZZanalysis(SEresNorm=resNorm,
                          DATAnorm=TRUE,
                          DataNumberCluster=NULL,
                          Membership=0.5,
                          Min.std=0.1,
                          Plot.Mfuzz=TRUE,
                          path.result=NULL)

```

---

MFUZZclustersNumber     *Automatic choice of the number of clusters to use for the Mfuzz analysis*

---

**Description**

The function uses `stats::kmeans()` or `FactoMineR::HCPC()` in order to compute the number of cluster for the `Mfuzz::mfuzz()` analysis.

**Usage**

```
MFUZZclustersNumber(
  SEresNorm,
  DATAnorm = TRUE,
  Method = "hcpc",
  Max.clust = 3,
  Min.std = 0.1,
  Plot.Cluster = TRUE,
  path.result = NULL
)
```

**Arguments**

<code>SEresNorm</code>	Results of the function <code>DATANormalization()</code> .
<code>DATAnorm</code>	TRUE or FALSE. TRUE as default. TRUE means the function uses the normalized data. FALSE means the function uses the raw counts data.
<code>Method</code>	"kmeans" or "hcpc". The method used for selecting the number of cluster to be used for the temporal cluster analysis (see Details). Method="kmeans" is advised for large number of genes.
<code>Max.clust</code>	Integer strictly superior to 1 indicating the maximum number of clusters. The default is <code>Max.clust=10</code> .
<code>Min.std</code>	Numeric positive value. All genes where their standard deviations are smaller than the threshold <code>Min.std</code> will be excluded.
<code>Plot.Cluster</code>	TRUE or FALSE. TRUE as default. If TRUE, the output graph will be plotted. Otherwise the graph will be plotted.
<code>path.result</code>	Character or NULL. Path to save the plot described in the section Value. If NULL, the graph will not be saved in a folder. NULL as default.

**Details**

All results are built from the results of our function `DATANormalization()`.

The `Mfuzz` package works with datasets where rows correspond to genes and columns correspond to times. If `RawCounts` (input of our function `DATAprepSE()`) contains several replicates per time, the algorithm computes the mean of replicates for each gene before using `Mfuzz::mfuzz()`. When there are several biological conditions, the algorithm realizes the `Mfuzz::mfuzz()` analysis for each biological condition.

The `kmeans` method or the hierarchical clustering method, respectively included in `stats::kmeans()` and `FactoMineR::HCPC()`, is used in order to compute the optimal number of clusters. If there are several biological conditions, the algorithm computes one optimal number of clusters per biological condition.

**Value**

The function returns the same SummarizedExperiment class object `SEresNorm` with the different elements below, saved in the metadata `Results[[1]][[4]]` of `SEresNorm`,

- the optimal number of clusters for each biological condition (between 2 and `Max.clust`).
- a data.frame with  $(N_{bc} + 1)$  columns and `Max.clust` rows with  $N_{bc}$  the number of biological conditions.
  - If `Method="kmeans"`, the *i*th rows and the *j*th column correspond to the within-cluster inertia (see `tot.withinss` from `stats::kmeans()`) dividing by the sum of the variance of each row of `ExprData` of the (*j*-1)th biological condition computed by `stats::kmeans()` with *i* clusters. When there is only one cluster, the within-cluster inertia corresponds to the sum of the variance of each row of `ExprData` (see `Details`). The first column contains integers between 1 and `Max.clust` which corresponds to the number of clusters selected for the `stats::kmeans()` analysis.
  - If `Method="hpcpc"`, the *j*th column correspond to the clustering heights (see the output height from `FactoMineR::HCPC()`) dividing by the maximum value of height. The first column contains integers between 1 and `Max.clust` which corresponds to the number of clusters selected for the `stats::kmeans()` analysis.
- a plot which gives
  - If `Method="kmeans"`, the evolution of the weighted within-cluster inertia per number of clusters (from 1 to `Max.clust`) for each biological condition. The optimal number of cluster for each biological condition will be colored in blue.
  - If `Method="hpcpc"`, the evolution of the scaled height per number of clusters (from 1 to `Max.clust`) for each biological condition. The optimal number of cluster for each biological condition will be colored in blue.

**See Also**

The function is called by `MFUZZanalysis()`.

**Examples**

```
## Data simulation
set.seed(33)
DATAclustSIM <- matrix(rnorm(12*10*3, sd=0.2,
                           mean=rep(c(rep(c(1, 6, 9, 4, 3, 1,
                                           6.5, 0.7, 10), times=2),
                                       rep(c(2, 3.6, 3.7, 5, 7.9, 8,
                                           7.5, 3.5, 3.4), times=2))),
                           each=10)),
                      nrow=30, ncol=12)
DATAclustSIM <- floor(DATAclustSIM*100)
##
colnames(DATAclustSIM) <- c("G1_t0_r1", "G1_t1_r1", "G1_t2_r1",
                          "G1_t0_r2", "G1_t1_r2", "G1_t2_r2",
                          "G2_t0_r3", "G2_t1_r3", "G2_t2_r3",
                          "G2_t0_r4", "G2_t1_r4", "G2_t2_r4")
##-----##
## Plot the temporal expression of each individual
```

```

graphics::matplot(t(rbind(DATAclustSIM[, 1:3], DATAclustSIM[, 4:6],
                        DATAclustSIM[, 7:9], DATAclustSIM[, 10:12])),
                 col=rep(c("black", "red"), each=6*10),
                 xlab="Time", ylab="Gene expression", type=c("b"), pch=19)

##-----##
## Preprocessing step
DATAclustSIM <- data.frame(DATAclustSIM)

resDATAprepSE <- DATAprepSE(RawCounts=DATAclustSIM,
                             Column.gene=NULL,
                             Group.position=1,
                             Time.position=2,
                             Individual.position=3)

## Normalization
resNorm <- DATAnormalization(Seres=resDATAprepSE,
                              Normalization="rle",
                              Plot.Boxplot=FALSE,
                              Colored.By.Factors=FALSE)

##-----##
resMFUZZcluster <- MFUZZclustersNumber(SeresNorm=resNorm,
                                       DATAnorm=FALSE,
                                       Method="hpc",
                                       Max.clust=5,
                                       Plot.Cluster=TRUE,
                                       path.result=NULL)

```

---

PCAanalysis

*Automatic PCA analysis (Main function)*


---

## Description

The functions performs an automatic principal component analysis (PCA) from a gene expression dataset where samples can belong to different biological conditions and/or time points.

## Usage

```

PCAanalysis(
  SEresNorm,
  DATAnorm = TRUE,
  gene.deletion = NULL,
  sample.deletion = NULL,
  Plot.PCA = TRUE,
  Mean.Accross.Time = FALSE,
  Color.Group = NULL,
  Phi = 25,
  Theta = 140,
  epsilon = 0.2,

```

```

Cex.point = 0.7,
Cex.label = 0.7,
motion3D = FALSE,
path.result = NULL,
Name.folder.pca = NULL
)

```

## Arguments

SEresNorm	Results of the function <a href="#">DATAnormalization()</a> .
DATAnorm	TRUE or FALSE. TRUE as default. TRUE means the function uses the normalized data. FALSE means the function uses the raw counts data.
gene.deletion	NULL or a vector of characters or a vector of integers. NULL as default. If <code>gene.deletion</code> is a vector of characters, all genes with names in <code>gene.deletion</code> will be deleted from the data set as input <code>RawCounts</code> of our function <a href="#">DATAprepSE()</a> . If <code>gene.deletion</code> is a vector of integers, all the corresponding row numbers will be deleted from the data set as input <code>RawCounts</code> of our function <a href="#">DATAprepSE()</a> . If <code>gene.deletion=NULL</code> all genes will be used in the construction of the PCA.
sample.deletion	NULL or a vector of characters or a vector of integers. NULL as default. If <code>sample.deletion</code> is a vector of characters, all samples with names in <code>sample.deletion</code> will not be used in the construction of the PCA. If <code>sample.deletion</code> is a vector of integers, all the corresponding column numbers will not be used in the construction of the PCA from the data set as input <code>RawCounts</code> of our function <a href="#">DATAprepSE()</a> . If <code>sample.deletion=NULL</code> all samples will be used in the construction of the PCA.
Plot.PCA	TRUE or FALSE. TRUE as default. If TRUE, PCA graphs will be plotted. Otherwise no graph will be plotted.
Mean.Accross.Time	TRUE or FALSE. FALSE as default. If FALSE and if <code>Time.position</code> (input of <a href="#">DATAprepSE()</a> ) is not set as NULL, consecutive time points within a sample are linked to help visualization of temporal patterns. If TRUE and if <code>Time.position</code> is not set as NULL, the mean per time of all genes is computed for each biological condition and the means of consecutive time points within biological condition are linked to help visualization of temporal patterns.
Color.Group	NULL or a data.frame with $N_{bc}$ rows and two columns where $N_{bc}$ is the number of biological conditions. If <code>Color.Group</code> (input of <a href="#">DATAprepSE()</a> ) is a data.frame, the first column must contain the name of each biological condition and the second column must contain the colors associated to each biological condition. If <code>Color.Group=NULL</code> , the function will automatically attribute a color for each biological condition. If samples belong to different time points only, <code>Color.Group</code> will not be used.
Phi	Angle defining the colatitude direction for the 3D PCA plot (see Details in <a href="#">graphics::persp()</a> ).
Theta	Angle defining the azimuthal direction for the 3D PCA plot (see Details in <a href="#">graphics::persp()</a> ).



<code>epsilon</code>	Non negative numeric value giving the length between points and their labels in all PCA plots which are not automatically plotted by <code>FactoMineR::PCA()</code> .
<code>Cex.point</code>	Non negative numeric value giving the size of points in all PCA plots which are not automatically plotted by <code>FactoMineR::PCA()</code> .
<code>Cex.label</code>	Non negative numeric value giving the size of the labels associated to each point of the all PCA graphs which are not automatically plotted by <code>FactoMineR::PCA()</code> .
<code>motion3D</code>	TRUE or FALSE. If TRUE, the 3D PCA plots will also be plotted in a rgl window (see <code>plot3Drgl::plotrgl()</code> ) allowing to interactively rotate and zoom.
<code>path.result</code>	Character or NULL. Path to save all results. If <code>path.result</code> contains a sub folder entitled "1_UnsupervisedAnalysis_Name.folder.pca" and a sub sub folder, "1-2_PCAanalysis_Name.folder.pca" all results will be saved in the sub folder "1_UnsupervisedAnalysis_Name.folder.pca/1-2_PCAanalysis_Name.folder.pca". Otherwise, a sub folder entitled "1_UnsupervisedAnalysis_Name.folder.pca" and/or a sub sub folder "1-2_PCAanalysis_Name.folder.pca" will be created in <code>path.result</code> and all results will be saved in "1_UnsupervisedAnalysis_Name.folder.pca/1-2_PCAanalysis_Name.folder.pca". If NULL, the results will not be saved in a folder. NULL as default.
<code>Name.folder.pca</code>	Character or NULL. If <code>Name.folder.pca</code> is a character, the folder and sub folder names which will contain the PCA graphs will respectively be "1_Unsupervised-Analysis_Name.folder.pca" and "1-2_PCAanalysis_Name.folder.pca". Otherwise, the folder and sub folder names will respectively be "1_Unsupervised-Analysis" and "1-2_PCAanalysis".

## Details

All results are built from the results of our function `DATAnormalization()`.

## Value

The function returns the same `SummarizedExperiment` class object `SEresNorm` with the outputs from the function `FactoMineR::PCA()`, and several 2D and 3D PCA graphs depending on the experimental design (if `Plot.PCA=TRUE`), saved in the metadata `Results[[1]][[2]]` of `SEresNorm`,

- When samples belong only to different biological conditions, the function returns a 2D and two 3D PCA graphs. In each graph, samples are colored with different colors for different biological conditions. The two 3D PCA graphs are identical but one of them will be opened in a rgl window (see `plot3Drgl::plotrgl()`) and it allows to interactively rotate and zoom.
- When samples belong only to different time points, the function returns
  - One 2D PCA graph, one 3D PCA graph and the same 3D PCA graph in a rgl window where samples are colored with different colors for different time points. Furthermore, lines are drawn between each pair of consecutive points for each sample (if `Mean.Accross.Time=FALSE`, otherwise it will be only between means).
  - The same graphs describe above but without lines.
- When samples belong to different time points and different biological conditions, the function returns

- One 2D PCA graph, one 3D PCA graph and the same 3D PCA graph in a rgl window where samples are colored with different colors for different time points. Furthermore, lines are drawn between each pair of consecutive points for each sample (if `Mean.Accross.Time=FALSE`, otherwise it will be only between means).
- The same graphs describe above but without lines.
- The same six following graphs for each biological condition (one PCA analysis per biological condition). One 2D PCA graph, one 3D PCA graph and the same 3D PCA graph in a rgl window where samples belong to only one biological condition and are colored with different colors for different time points. Furthermore, lines are drawn between each pair of consecutive points for each sample (if `Mean.Accross.Time=FALSE`, otherwise it will be only between means). The three others graphs are identical to the three previous ones but without lines.

The interactive 3D graphs will be plotted only if `motion3D=TRUE`.

### See Also

The function calls the R functions [PCAgraphics\(\)](#) and [ColnamesToFactors\(\)](#).

### Examples

```
## Simulation raw counts
resSIMcount <- RawCountsSimulation(Nb.Group=2, Nb.Time=3, Nb.per.GT=4,
                                   Nb.Gene=10)

## Preprocessing step
resDATAprepSE <- DATAprepSE(RawCounts=resSIMcount$Sim.dat,
                              Column.gene=1,
                              Group.position=1,
                              Time.position=2,
                              Individual.position=3)

## Normalization
resNorm <- DATAnormalization(SEres=resDATAprepSE,
                              Normalization="rle",
                              Plot.Boxplot=FALSE,
                              Colored.By.Factors=FALSE)

## Color for each group
GROUPcolor <- data.frame(Name=c("G1", "G2"), Col=c("black", "red"))
##-----##
resPCAanalysis <- PCAanalysis(SEresNorm=resNorm,
                              DATAnorm=TRUE,
                              gene.deletion=c("Gene1", "Gene5"),
                              sample.deletion=c(2, 6),
                              Plot.PCA=TRUE,
                              Mean.Accross.Time=FALSE,
                              Color.Group=GROUPcolor,
                              motion3D=FALSE,
                              Phi=25, Theta=140,
                              Cex.label=0.7, Cex.point=0.7, epsilon=0.2,
                              path.result=NULL, Name.folder.pca=NULL)
```

**Description**

The function plots 2D and 3D PCA using the function [PCArealization\(\)](#) which realizes a PCA analysis. This function is called repeatedly by the function [PCAanalysis\(\)](#) if samples belong to different biological conditions and time points.

**Usage**

```
PCAGraphics(
  SEresNorm,
  DATAnorm = TRUE,
  gene.deletion = NULL,
  sample.deletion = NULL,
  Plot.PCA = TRUE,
  Mean.Accross.Time = FALSE,
  Color.Group = NULL,
  motion3D = FALSE,
  Phi = 25,
  Theta = 140,
  epsilon = 0.2,
  Cex.point = 0.7,
  Cex.label = 0.7,
  path.result = NULL,
  Name.file.pca = NULL
)
```

**Arguments**

<code>SEresNorm</code>	Results of the function <a href="#">DATAnormalization()</a> .
<code>DATAnorm</code>	TRUE or FALSE. TRUE as default. TRUE means the function uses the normalized data. FALSE means the function uses the raw counts data.
<code>gene.deletion</code>	NULL or a vector of characters or a vector of integers. NULL as default. If <code>gene.deletion</code> is a vector of characters, all genes with names in <code>gene.deletion</code> will be deleted from the data set as input <code>RawCounts</code> of our function <a href="#">DATAprepSE()</a> . If <code>gene.deletion</code> is a vector of integers, all the corresponding row numbers will be deleted from the data set as input <code>RawCounts</code> of our function <a href="#">DATAprepSE()</a> . If <code>gene.deletion=NULL</code> all genes will be used in the construction of the PCA.
<code>sample.deletion</code>	NULL or a vector of characters or a vector of integers. NULL as default. If <code>sample.deletion</code> is a vector of characters, all samples with names in <code>sample.deletion</code> will not be used in the construction of the PCA. If <code>sample.deletion</code> is a vector of integers, all the corresponding column numbers will not be used in the construction of the PCA from the data set as input <code>RawCounts</code> of our function

	<code>DATAprepSE()</code> . If <code>sample.deletion=NULL</code> all samples will be used in the construction of the PCA.
<code>Plot.PCA</code>	TRUE or FALSE. TRUE as default. If TRUE, PCA graphs will be plotted. Otherwise no graph will be plotted.
<code>Mean.Accross.Time</code>	TRUE or FALSE. FALSE as default. If FALSE and if <code>Time.position</code> (input of <code>DATAprepSE()</code> ) is not set as NULL, consecutive time points within a sample are linked to help visualization of temporal patterns. If TRUE and if <code>Time.position</code> is not set as NULL, the mean per time of all genes is computed for each biological condition and the means of consecutive time points within biological condition are linked to help visualization of temporal patterns.
<code>Color.Group</code>	NULL or a data.frame with $N_{bc}$ rows and two columns where $N_{bc}$ is the number of biological conditions. If <code>Color.Group</code> (input of <code>DATAprepSE()</code> ) is a data.frame, the first column must contain the name of each biological condition and the second column must contain the colors associated to each biological condition. If <code>Color.Group=NULL</code> , the function will automatically attribute a color for each biological condition. If samples belong to different time points only, <code>Color.Group</code> will not be used.
<code>motion3D</code>	TRUE or FALSE. If TRUE, the 3D PCA plots will also be plotted in a rgl window (see <code>plot3Drgl::plotrgl()</code> ) allowing to interactively rotate and zoom.
<code>Phi</code>	Angle defining the colatitude direction for the 3D PCA plot (see Details in <code>graphics::persp()</code> ).
<code>Theta</code>	Angle defining the azimuthal direction for the 3D PCA plot (see Details in <code>graphics::persp()</code> ).
<code>epsilon</code>	Non negative numeric value giving the length between points and their labels in all PCA plots which are not automatically plotted by <code>FactoMineR::PCA()</code> .
<code>Cex.point</code>	Non negative numeric value giving the size of points in all PCA plots which are not automatically plotted by <code>FactoMineR::PCA()</code> .
<code>Cex.label</code>	Non negative numeric value giving the size of the labels associated to each point of the all PCA graphs which are not automatically plotted by <code>FactoMineR::PCA()</code> .
<code>path.result</code>	Character or NULL. Path to save the different PCA graphs. If NULL, the different PCA graphs will not be saved in a folder. NULL as default.
<code>Name.file.pca</code>	Character or NULL. If <code>Name.file.pca</code> is a character, <code>Name.file.pca</code> will be added at the beginning of all names of the saved graphs.

### Details

All results are built from the results of our function `DATANormalization()`.

### Value

The function returns the same `SummarizedExperiment` class object `SEResNorm` with the outputs from the function `FactoMineR::PCA()`, and plots several 2D and 3D PCA graphs depending on the experimental design (if `Plot.PCA=TRUE`), saved in the metadata `Results[[1]][[2]]` of `SEResNorm`,

- When samples belong only to different biological conditions, the function returns a 2D and two 3D PCA graphs. In each graph, samples are colored with different colors for different biological conditions. The two 3D PCA graphs are identical but one of them will be opened in a rgl window (see `plot3Drgl::plotrgl()`) and it allows to interactively rotate and zoom.
- When samples belong only to different time points, the function returns
  - One 2D PCA graph, one 3D PCA graph and the same 3D PCA graph in a rgl window where samples are colored with different colors for different time points. Furthermore, lines are drawn between each pair of consecutive points for each sample (if `Mean.Accross.Time=FALSE`, otherwise it will be only between means).
  - The same graphs describe above but without lines.
- When samples belong to different time points and different biological conditions, the function returns
  - One 2D PCA graph, one 3D PCA graph and the same 3D PCA graph in a rgl window where samples are colored with different colors for different time points. Furthermore, lines are drawn between each pair of consecutive points for each sample (if `Mean.Accross.Time=FALSE`, otherwise it will be only between means).
  - The same graphs describe above but without lines.
  - The same six following graphs for each biological condition (one PCA analysis per biological condition). One 2D PCA graph, one 3D PCA graph and the same 3D PCA graph in a rgl window where samples belong to only one biological condition and are colored with different colors for different time points. Furthermore, lines are drawn between each pair of consecutive points for each sample (if `Mean.Accross.Time=FALSE`, otherwise it will be only between means). The three others graphs are identical to the three previous ones but without lines.

The interactive 3D graphs will be plotted only if `motion3D=TRUE`.

### See Also

This function is called by our function `PCAanalysis()` and calls our function `PCArealization()`.

### Examples

```
## Simulation raw counts
resSIMcount <- RawCountsSimulation(Nb.Group=2, Nb.Time=3, Nb.per.GT=4,
                                   Nb.Gene=10)

## Preprocessing step
resDATAprepSE <- DATAprepSE(RawCounts=resSIMcount$Sim.dat,
                              Column.gene=1,
                              Group.position=1,
                              Time.position=2,
                              Individual.position=3)

## Normalization
resNorm <- DATAnormalization(SEres=resDATAprepSE,
                              Normalization="rle",
                              Plot.Boxplot=FALSE,
                              Colored.By.Factors=FALSE)

## Color for each group
GROUPcolor <- data.frame(Name=c("G1", "G2"), Col=c("black", "red"))
```

```
##-----##
resPCAgraph <- PCAgraphics(SEresNorm=resNorm,
                           DATAnorm=TRUE,
                           gene.deletion=c("Gene1", "Gene5"),
                           sample.deletion=c(2,6),
                           Plot.PCA=TRUE,
                           Mean.Accross.Time=FALSE,
                           Color.Group=GROUPcolor,
                           motion3D=FALSE,
                           Phi=25, Theta=140, Cex.label=0.7,
                           Cex.point=0.7, epsilon=0.2,
                           path.result=NULL, Name.file.pca=NULL)
```

---

PCAp preprocessing

*Reshaped dataset for factorial analysis.*


---

## Description

The function generates a SummarizedExperiment class object containing the dataset reshaped from the original dataset, to be used by the function `FactoMineR::PCA()`, which performs the Principal Component Analysis (PCA). This function is called by the function `PCArealization()`, which also calls the function `FactoMineR::PCA()`.

## Usage

```
PCAp preprocessing(SEresNorm, DATAnorm = TRUE)
```

## Arguments

SEresNorm	Results of the function <code>DATAnormalization()</code> .
DATAnorm	TRUE or FALSE. TRUE as default. TRUE means the function uses the normalized data. FALSE means the function uses the raw counts data.

## Details

All results are built from the results of our function `DATAnormalization()`.

## Value

The function returns the same SummarizedExperiment class object SEresNorm with the different elements below

- information for the functions `PCArealization()` and `PCAgraphics()`
- a reshape of the originally dataset for the PCA analysis (realized by the function `PCArealization()`)

saved in the metadata `Results[[1]][[2]]` of SEresNorm.

The reshaped dataset which corresponds to a data.frame with  $(N_g + k)$  columns and  $N_s$  rows, where  $N_g$  is the number of genes,  $N_s$  is the number of samples and

- $k = 1$  if samples belong to different biological condition or time points. In that case, the first column will contain the biological condition or the time point associated to each sample.
- $k = 2$  if samples belong to different biological condition and time points. In that case, the first column will contain the biological condition and the second column the time point associated to each sample.

The other  $N_g$  columns form a sub data.frame which is a transpose of the data.frame composed of the  $N_s$  numeric columns of ExprData.

### See Also

The function is called by our function [PCArealization\(\)](#) and uses our function [DATAnormalization\(\)](#).

### Examples

```
## Simulation raw counts
resSIMcount <- RawCountsSimulation(Nb.Group=2, Nb.Time=3, Nb.per.GT=4,
                                   Nb.Gene=10)

## Preprocessing step
resDATAprepSE <- DATAprepSE(RawCounts=resSIMcount$Sim.dat,
                              Column.gene=1,
                              Group.position=1,
                              Time.position=2,
                              Individual.position=3)

## Normalization
resNorm <- DATAnormalization(SEres=resDATAprepSE,
                              Normalization="rle",
                              Plot.Boxplot=FALSE,
                              Colored.By.Factors=FALSE)

##-----##
resPCadata <- PCApreprocessing(SEresNorm=resNorm,
                               DATAnorm=TRUE)
```

---

PCArealization

*PCA realization*

---

### Description

From a gene expression dataset, the functions performs the Principal Component Analysis (PCA) through the R function [FactoMineR::PCA\(\)](#).

### Usage

```
PCArealization(
  SEresNorm,
  DATAnorm = TRUE,
  gene.deletion = NULL,
  sample.deletion = NULL,
  Supp.del.sample = FALSE
)
```

## Arguments

<code>SEresNorm</code>	Results of the function <code>DATAnormalization()</code> .
<code>DATAnorm</code>	TRUE or FALSE. TRUE as default. TRUE means the function uses the normalized data. FALSE means the function uses the raw counts data.
<code>gene.deletion</code>	NULL or a vector of characters or a vector of integers. NULL as default. If <code>gene.deletion</code> is a vector of characters, all genes with names in <code>gene.deletion</code> will be deleted from the data set as input <code>RawCounts</code> of our function <code>DATAprepSE()</code> . If <code>gene.deletion</code> is a vector of integers, all the corresponding row numbers will be deleted from the data set as input <code>RawCounts</code> of our function <code>DATAprepSE()</code> . If <code>gene.deletion=NULL</code> all genes will be used in the construction of the PCA.
<code>sample.deletion</code>	NULL or a vector of characters or a vector of integers. NULL as default. If <code>sample.deletion</code> is a vector of characters, all samples with names in <code>sample.deletion</code> will not be used in the construction of the PCA. If <code>sample.deletion</code> is a vector of integers, all the corresponding column numbers will not be used in the construction of the PCA from the data set as input <code>RawCounts</code> of our function <code>DATAprepSE()</code> . If <code>sample.deletion=NULL</code> all samples will be used in the construction of the PCA.
<code>Supp.del.sample</code>	TRUE or FALSE. FALSE by default. If FALSE, the samples selected with <code>sample.deletion</code> will be deleted. If TRUE, the samples selected with <code>sample.deletion</code> will be plotted. These individuals are called supplementary individuals in <code>FactoMineR::PCA()</code> .

## Details

All results are built from the results of our function `DATAnormalization()`.

## Value

The function returns the same `SummarizedExperiment` class object `SEresNorm` but with the output of the `FactoMineR::PCA()` function (see `FactoMineR::PCA()`) saved in the metadata `Results[[1]][[2]]` of `SEresNorm`.

## See Also

The `PCArealization()` function

- is used by the following functions of our package : `PCAanalysis()` and `HCPCanalysis()`.
- calls the R function `PCApreprocessing()` for reshaping the data and uses its output for performing a Principal Component (PCA) with `FactoMineR::PCA()`.

## Examples

```
## Simulation raw counts
resSIMcount <- RawCountsSimulation(Nb.Group=2, Nb.Time=3, Nb.per.GT=4,
                                   Nb.Gene=10)

## Preprocessing step
resDATAprepSE <- DATAprepSE(RawCounts=resSIMcount$Sim.dat,
```



```

        Column.gene=1,
        Group.position=1,
        Time.position=2,
        Individual.position=3)

## Normalization
resNorm <- DATAnormalization(SEres=resDATAprepSE,
                              Normalization="rle",
                              Plot.Boxplot=FALSE,
                              Colored.By.Factors=FALSE)

##-----##
resPCAex <- PCArealization(SEresNorm=resNorm,
                          DATAnorm=TRUE,
                          gene.deletion=c(3, 5),
                          sample.deletion=c("G1_t0_Ind2", "G1_t1_Ind3"),
                          Supp.del.sample=FALSE)

##-----##
resPCAex2 <- PCArealization(SEresNorm=resNorm,
                           DATAnorm=TRUE,
                           gene.deletion=c("Gene3", "Gene5"),
                           sample.deletion=c(3, 8),
                           Supp.del.sample=TRUE)

```

---

RawCountsSimulation    *RNA-seq raw counts data simulation*

---

## Description

The function simulates an in silico RNA-seq raw counts data inspired from the model used in the DESeq2 package. It is used in some examples of other functions.

## Usage

```
RawCountsSimulation(Nb.Group, Nb.Time, Nb.per.GT, Nb.Gene)
```

## Arguments

Nb.Group	Non negative integer. Number of biological condition (minimum 1).
Nb.Time	Non negative integer. Number of time points (minimum 1).
Nb.per.GT	Non negative integer. Number of sample for each condition and time (minimum 1).
Nb.Gene	Non negative integer. Number of genes (minimum 1)

## Value

A simulated RNA-seq raw counts data.

## Examples

```
RawCountsSimulation(Nb.Group=3, Nb.Time=5, Nb.per.GT=7, Nb.Gene=50)
## RawCountsSimulation(Nb.Group=1, Nb.Time=5, Nb.per.GT=7, Nb.Gene=50)
## RawCountsSimulation(Nb.Group=3, Nb.Time=1, Nb.per.GT=7, Nb.Gene=50)
```

---

```
RawCounts_Antoszewski2022_MOUSEsub500
      Mouse raw counts data
```

---

## Description

There are 4 groups : samples with or without hyper activation of the gene NOTTCH1 (N1ha versus N1wt) and with or without knock out of the gene TCF1 (T1ko versus T1wt). The original dataset has 39017 genes but we kept only 500 genes in order to increase the speed of each function in our algorithm.

## Usage

```
data(RawCounts_Antoszewski2022_MOUSEsub500)
```

## Format

A data frame with 500 rows (genes) and 13 columns (samples). The column names are as follow

**Gene** ENSEMBL gene names.

**N1wtT1wt\_r1** The sample is the first replica (r1) of the biological condition N1wt and T1wt.

**N1wtT1wt\_r2** The sample is the second replica (r2) of the biological condition N1wt and T1wt.

**N1wtT1wt\_r3** The sample is the third replica (r3) of the biological condition N1wt and T1wt.

**N1haT1wt\_r4** The sample is the first replica (r4) of the biological condition N1ha and T1wt.

**N1haT1wt\_r5** The sample is the second replica (r5) of the biological condition N1ha and T1wt.

**N1haT1wt\_r6** The sample is the third replica (r6) of the biological condition N1ha and T1wt.

**N1haT1ko\_r7** The sample is the first replica (r7) of the biological condition N1ha and T1ko.

**N1haT1ko\_r8** The sample is the second replica (r8) of the biological condition N1ha and T1ko.

**N1haT1ko\_r9** The sample is the third replica (r9) of the biological condition N1ha and T1ko.

**N1wtT1ko\_r10** The sample is the first replica (r10) of the biological condition N1wt and T1ko.

**N1wtT1ko\_r11** The sample is the second replica (r11) of the biological condition N1wt and T1ko.

**N1wtT1ko\_r12** The sample is the third replica (r12) of the biological condition N1wt and T1ko.

## Details

The following is quoted from the GEO series GSE169116 (link in source):

Summary : "NOTCH1 is a well-established lineage specifier for T cells and among the most frequently mutated genes throughout all subclasses of T cell acute lymphoblastic leukemia (T-ALL). How oncogenic NOTCH1 signaling launches a leukemia-prone chromatin landscape during T-ALL initiation is unknown. Here we demonstrate an essential role for the high-mobility-group transcription factor Tcf1 in orchestrating chromatin accessibility and topology allowing for aberrant Notch1 signaling to convey its oncogenic function. Although essential, Tcf1 is not sufficient to initiate leukemia. The formation of a leukemia-prone landscape at the distal Notch1-regulated Myc enhancer, which is fundamental to this disease, is Tcf1-dependent and occurs within the earliest progenitor stage even before cells adopt a T lymphocyte or leukemic fate. Moreover, we discovered an additional evolutionarily conserved Tcf1-regulated enhancer element, in the distal Myc-enhancer, which is important for the transition of pre-leukemic cells to full-blown disease."

Overall design: "Expression profile comparisons of sorted LSK derived from C57BL/6J; Sv/129 compound mice with Notch1 induced or Tcf1 knocked-down."

We kept 500 genes only in order to increase the speed for each example.

## Value

Mouse dataset with four biological conditions.

## Source

This dataset comes from Gene Expression Omnibus (GEO) <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE169116>. The name of the samples was renamed in order to be used with our package.

## References

Antoszewski M, Fournier N, Ruiz Buendía GA, Lourenco J et al. 'Tcf1 is essential for initiation of oncogenic Notch1-driven chromatin topology in T-ALL'. Blood 2022 Jan 12. PMID:35020836. GEO:GSE169116.

## Examples

```
data(RawCounts_Antoszewski2022_MOUSEsub500)
```

---

RawCounts\_Leong2014\_FISSIONsub500wt

*Yeast times series raw counts data after stimulation with and without silencing*

---

## Description

Raw counts data for fission yeast RNA-Seq experiment with two groups (wt and mut), 6 times (0, 15min, 30min, 60min, 120min, 180min) and 3 replicates for each group and time. The original dataset has 7039 genes but we kept only 500 genes in order to increase the speed of each function in our algorithm.

**Usage**

```
data(RawCounts_Leong2014_FISSIONsub500wt)
```

**Format**

A data frame with 500 rows (genes) and 37 columns (samples). The column names are as follow

**Gene** Gene name

**wt\_t0\_r1** The sample is the first replica (r1) of the biological condition control (wt) at time t0 (0 min)

**wt\_t0\_r2** The sample is the second replica (r2) of the biological condition control (wt) at time t0 (0 min)

**wt\_t0\_r3** The sample is the third replica (r3) of the biological condition control (wt) at time t0 (0 min)

**wt\_t1\_r1** The sample is the first replica (r1) of the biological condition control (wt) at time t1 (15 min)

**wt\_t1\_r2** The sample is the second replica (r2) of the biological condition control (wt) at time t1 (15 min)

**wt\_t1\_r3** The sample is the third replica (r3) of the biological condition control (wt) at time t1 (15 min)

**wt\_t2\_r1** The sample is the first replica (r1) of the biological condition control (wt) at time t2 (30 min)

**wt\_t2\_r2** The sample is the second replica (r2) of the biological condition control (wt) at time t2 (30 min)

**wt\_t2\_r3** The sample is the third replica (r3) of the biological condition control (wt) at time t2 (30 min)

**wt\_t3\_r1** The sample is the first replica (r1) of the biological condition control (wt) at time t3 (60 min)

**wt\_t3\_r2** The sample is the second replica (r2) of the biological condition control (wt) at time t3 (60 min)

**wt\_t3\_r3** The sample is the third replica (r3) of the biological condition control (wt) at time t3 (60 min)

**wt\_t4\_r1** The sample is the first replica (r1) of the biological condition control (wt) at time t4 (120 min)

**wt\_t4\_r2** The sample is the second replica (r2) of the biological condition control (wt) at time t4 (120 min)

**wt\_t4\_r3** The sample is the third replica (r3) of the biological condition control (wt) at time t4 (120 min)

**wt\_t5\_r1** The sample is the first replica (r1) of the biological condition control (wt) at time t5 (180 min)

**wt\_t5\_r2** The sample is the second replica (r2) of the biological condition control (wt) at time t5 (180 min)

**wt\_t5\_r3** The sample is the third replica (r3) of the biological condition control (wt) at time t5 (180 min)

## Details

The following is quoted from the GEO series GSE56761 (link in source):

Summary: "Mitogen Activated Protein Kinase (MAPK) signaling cascades transduce information arising from events external to the cell, such as environmental stresses, to a variety of downstream effectors and transcription factors. The fission yeast stress activated MAP kinase (SAPK) pathway is conserved with the p38 and JNK pathways in humans, and comprises the MAPKKKs Win1, Wis4, the MAPKK Wis1, and the MAPK, Sty1. Sty1 and its main downstream effector Atf1 regulate a large set of core environmental stress response genes. The fission yeast genome encodes three other ATF proteins: Atf21, Atf31 and Pcr1. Among these, atf21 is specifically induced under conditions of high osmolarity. We have therefore instigated a programme to investigate the role played by non coding RNAs (ncRNAs) in response to osmotic stress challenge in wild type and atf21Delta cells. By integrating global proteomics and RNA sequencing data, we identified a systematic program in which elevated antisense RNAs arising both from ncRNAs and from 3'-overlapping convergent gene-pairs is directly associated with substantial reductions in protein levels throughout the fission yeast genome. We also found an extensive array of ncRNAs with trans associations that have the potential to influence different biological processes and stress responses in fission yeast, suggesting ncRNAs comprise additional components of the SAPK regulatory system".

Overall design: "Global transcription profiles of fission yeast wild type (WT) and atf21del strains over an osmotic stress time course following treatment with 1M sorbitol at 0, 15, 30, 60, 120 and 180 mins. Strand-specific single end sequencing of total RNA was performed in biological triplicates on the Applied Biosystems SOLiD 5500xl Genetic Analyzer System".

We kept 500 genes only in order to increase the speed for each example.

## Value

Yeast dataset with 6 time measurements.

## Source

This dataset can be found in the R Package fission. <https://bioconductor.org/packages/release/data/experiment/html/fission.html> Link of GEO series GSE56761: <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE56761>. The name of the samples was renamed in order to be used with our package.

## References

Leong HS, Dawson K, Wirth C, Li Y et al. 'A global non-coding RNA system modulates fission yeast protein levels in response to stress'. Nat Commun 2014 May 23;5:3947. PMID:24853205. GEO:GSE56761.

## Examples

```
data(RawCounts_Leong2014_FISSIONsub500wt)
```

---

RawCounts\_Schleiss2021\_CLLsub500

*Human CCL times series raw counts data after stimulation with and without silencing*

---

## Description

This time series count data (read counts) represents the temporal transcriptional response of primary human chronic lymphocytic leukemia (CLL)-cells after B-cell receptor stimulation. There are 9 time points (before stimulation (0h) and at the time points 1h, 1h30, 3h30, 6h30, 12h, 24h, 48h and 96h after cell stimulation) and samples are divided in two groups : Proliferating (P) and Non Proliferating (NP). There are also 3 replicates for a time and biological condition. The original dataset has 25369 genes but we kept only 500 genes in order to increase the speed of each function in our algorithm.

## Usage

```
data(RawCounts_Schleiss2021_CLLsub500)
```

## Format

A data frame with 500 rows (genes) and 55 columns (samples). The column names are as follow

**Genes** Symbol gene name.

**CLL\_P\_r1\_t0** The sample is the first replica (r1) of the biological condition control (P) at time t0 (0h)

**CLL\_P\_r1\_t1** The sample is the first replica (r1) of the biological condition control (P) at time t1 (1h)

**CLL\_P\_r1\_t2** The sample is the first replica (r1) of the biological condition control (P) at time t2 (1h30)

**CLL\_P\_r1\_t3** The sample is the first replica (r1) of the biological condition control (P) at time t3 (3h30)

**CLL\_P\_r1\_t4** The sample is the first replica (r1) of the biological condition control (P) at time t4 (6h30)

**CLL\_P\_r1\_t5** The sample is the first replica (r1) of the biological condition control (P) at time t5 (12h)

**CLL\_P\_r1\_t6** The sample is the first replica (r1) of the biological condition control (P) at time t6 (24h)

**CLL\_P\_r1\_t7** The sample is the first replica (r1) of the biological condition control (P) at time t7 (48h)

**CLL\_P\_r1\_t8** The sample is the first replica (r1) of the biological condition control (P) at time t8 (96h)

**CLL\_P\_r2\_t0** The sample is the second replica (r2) of the biological condition control (P) at time t0 (0h)

- CLL\_P\_r2\_t1** The sample is the second replica (r2) of the biological condition control (P) at time t1 (1h)
- CLL\_P\_r2\_t2** The sample is the second replica (r2) of the biological condition control (P) at time t2 (1h30)
- CLL\_P\_r2\_t3** The sample is the second replica (r2) of the biological condition control (P) at time t3 (3h30)
- CLL\_P\_r2\_t4** The sample is the second replica (r2) of the biological condition control (P) at time t4 (6h30)
- CLL\_P\_r2\_t5** The sample is the second replica (r2) of the biological condition control (P) at time t5 (12h)
- CLL\_P\_r2\_t6** The sample is the second replica (r2) of the biological condition control (P) at time t6 (24h)
- CLL\_P\_r2\_t7** The sample is the second replica (r2) of the biological condition control (P) at time t7 (48h)
- CLL\_P\_r2\_t8** The sample is the second replica (r2) of the biological condition control (P) at time t8 (96h)
- CLL\_P\_r3\_t0** The sample is the third replica (r3) of the biological condition control (P) at time t0 (0h)
- CLL\_P\_r3\_t1** The sample is the third replica (r3) of the biological condition control (P) at time t1 (1h)
- CLL\_P\_r3\_t2** The sample is the third replica (r3) of the biological condition control (P) at time t2 (1h30)
- CLL\_P\_r3\_t3** The sample is the third replica (r3) of the biological condition control (P) at time t3 (3h30)
- CLL\_P\_r3\_t4** The sample is the third replica (r3) of the biological condition control (P) at time t4 (6h30)
- CLL\_P\_r3\_t5** The sample is the third replica (r3) of the biological condition control (P) at time t5 (12h)
- CLL\_P\_r3\_t6** The sample is the third replica (r3) of the biological condition control (P) at time t6 (24h)
- CLL\_P\_r3\_t7** The sample is the third replica (r3) of the biological condition control (P) at time t7 (48h)
- CLL\_P\_r3\_t8** The sample is the third replica (r3) of the biological condition control (P) at time t8 (96h)
- CLL\_NP\_r4\_t0** The sample is the first replica (r4) of the biological condition control (NP) at time t0 (0h)
- CLL\_NP\_r4\_t1** The sample is the first replica (r4) of the biological condition control (NP) at time t1 (1h)
- CLL\_NP\_r4\_t2** The sample is the first replica (r4) of the biological condition control (NP) at time t2 (1h30)
- CLL\_NP\_r4\_t3** The sample is the first replica (r4) of the biological condition control (NP) at time t3 (3h30)

- CLL\_NP\_r4\_t4** The sample is the first replica (r4) of the biological condition control (NP) at time t4 (6h30)
- CLL\_NP\_r4\_t5** The sample is the first replica (r4) of the biological condition control (NP) at time t5 (12h)
- CLL\_NP\_r4\_t6** The sample is the first replica (r4) of the biological condition control (NP) at time t6 (24h)
- CLL\_NP\_r4\_t7** The sample is the first replica (r4) of the biological condition control (NP) at time t7 (48h)
- CLL\_NP\_r4\_t8** The sample is the first replica (r4) of the biological condition control (NP) at time t8 (96h)
- CLL\_NP\_r5\_t0** The sample is the second replica (r5) of the biological condition control (NP) at time t0 (0h)
- CLL\_NP\_r5\_t1** The sample is the second replica (r5) of the biological condition control (NP) at time t1 (1h)
- CLL\_NP\_r5\_t2** The sample is the second replica (r5) of the biological condition control (NP) at time t2 (1h30)
- CLL\_NP\_r5\_t3** The sample is the second replica (r5) of the biological condition control (NP) at time t3 (3h30)
- CLL\_NP\_r5\_t4** The sample is the second replica (r5) of the biological condition control (NP) at time t4 (6h30)
- CLL\_NP\_r5\_t5** The sample is the second replica (r5) of the biological condition control (NP) at time t5 (12h)
- CLL\_NP\_r5\_t6** The sample is the second replica (r5) of the biological condition control (NP) at time t6 (24h)
- CLL\_NP\_r5\_t7** The sample is the second replica (r5) of the biological condition control (NP) at time t7 (48h)
- CLL\_NP\_r5\_t8** The sample is the second replica (r5) of the biological condition control (NP) at time t8 (96h)
- CLL\_NP\_r6\_t0** The sample is the third replica (r6) of the biological condition control (NP) at time t0 (0h)
- CLL\_NP\_r6\_t1** The sample is the third replica (r6) of the biological condition control (NP) at time t1 (1h)
- CLL\_NP\_r6\_t2** The sample is the third replica (r6) of the biological condition control (NP) at time t2 (1h30)
- CLL\_NP\_r6\_t3** The sample is the third replica (r6) of the biological condition control (NP) at time t3 (3h30)
- CLL\_NP\_r6\_t4** The sample is the third replica (r6) of the biological condition control (NP) at time t4 (6h30)
- CLL\_NP\_r6\_t5** The sample is the third replica (r6) of the biological condition control (NP) at time t5 (12h)
- CLL\_NP\_r6\_t6** The sample is the third replica (r6) of the biological condition control (NP) at time t6 (24h)



**CLL\_NP\_r6\_t7** The sample is the third replica (r6) of the biological condition control (NP) at time t7 (48h)

**CLL\_NP\_r6\_t8** The sample is the third replica (r6) of the biological condition control (NP) at time t8 (96h)

## Details

The following is quoted from the GEO series GSE130385 (link in source):

Summary: "The B-cell receptor (BCR) signaling is crucial for the pathophysiology of most leukemias and lymphomas originated from mature B lymphocytes and has emerged as a new therapeutic target, especially for chronic lymphocytic leukemia (CLL). However, the precise mechanisms by which BCR signaling controls neoplastic B-cell proliferation are ill characterized. This work was performed using primary leukemic cells of untreated patients at initial stage of CLL (Binet stage A / Rai 0) presenting biological characteristics of aggressive form of the disease (unmutated IGHV genes and ZAP70 protein expression). In order to mimic the primary leukemogenic step occurring in vivo, this study focused on the BCR-dependent proliferation of CLL cells induced ex vivo using anti-IgM, together with mandatory co-stimulating factors (CD40L, IL-4 and IL-21) (Schleiss, Sci Rep, 2019). Cell proliferation was objectivized by the emergence of proliferative clusters and the presence of more than 25% of CLL cells that did undergo division within the cell culture at day 6. To capture the specific actors of the proliferative response in these samples, we also included non-proliferating control CLL samples. Gene expression was analyzed by RNA-seq before stimulation (T0) and at the time points 1h, 1h30, 3h30, 6h30, 12h, 24h, 48h and 96h after cell stimulation (n=54 data points), the latest time points corresponding to the emergence of the proliferation clusters."

Overall design: "Temporal transcriptional response (T0 + 8 time points) of primary chronic lymphocytic leukemia (CLL) cells after BCR engagement ex vivo (anti-IgM, IL-4, CD40Ligand and IL-21) of 3 Proliferating (P1, P2, P3) and 3 Non Proliferating samples (NP1, NP2, NP3)".

## Value

Human CCL times series dataset with two biological conditions and with 9 time measurements.

## Source

This dataset comes from Gene Expression Omnibus (GEO) <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE130385>. I rewrite the name of the sample in order to be used with my package.

## References

Schleiss C, Carapito R, Fornecker LM, Muller L et al. 'Temporal multiomic modeling reveals a B-cell receptor proliferative program in chronic lymphocytic leukemia'. *Leukemia* 2021 May;35(5):1463-1474. PMID:33833385. GEO:GSE130385

## Examples

```
data(RawCounts_Schleiss2021_CLLsub500)
```

---

 RawCounts\_Weger2021\_MOUSEsub500

*Mouse count data with four biological conditions, six time measurements and 500 genes.*

---

## Description

This time series count data (read counts) represents the temporal transcriptional response (six time measurements across the course of a day) of Bmal1 wild-type (WT) and Cry1/2 WT, Bmal1 KO and Cry1/2 WT, Bmal1 (WT) and Cry1/2 KO, and Bmal1 KO and Cry1/2 KO mice under an ad libitum (AL) or night restricted feeding (RF) regimen. Therefore, there are eight biological conditions. As there are only two mice per biological condition, we will not consider the effect of the regimen. The original dataset has 40327 genes but we kept only 500 genes in order to increase the speed of each function in our algorithm.

## Usage

```
data(RawCounts_Weger2021_MOUSEsub500)
```

## Format

A data frame with 500 rows (genes) and 97 columns (samples). The column names are as follow

**Gene** ENSEMBL gene names.

**BmKo\_t0\_r1** The sample is the first replica (r1) of the biological condition Bmal1 and KO at time t0 (00h).

**BmKo\_t1\_r1** The sample is the first replica (r1) of the biological condition Bmal1 and KO at time t1 (04h).

**BmKo\_t2\_r1** The sample is the first replica (r1) of the biological condition Bmal1 and KO at time t2 (08h).

**BmKo\_t3\_r1** The sample is the first replica (r1) of the biological condition Bmal1 and KO at time t3 (12h).

**BmKo\_t4\_r1** The sample is the first replica (r1) of the biological condition Bmal1 and KO at time t4 (16h).

**BmKo\_t5\_r1** The sample is the first replica (r1) of the biological condition Bmal1 and KO at time t5 (20h).

**BmKo\_t0\_r2** The sample is the first replica (r2) of the biological condition Bmal1 and KO at time t0 (00h).

**BmKo\_t1\_r2** The sample is the first replica (r2) of the biological condition Bmal1 and KO at time t1 (04h).

**BmKo\_t2\_r2** The sample is the first replica (r2) of the biological condition Bmal1 and KO at time t2 (08h).

**BmKo\_t3\_r2** The sample is the first replica (r2) of the biological condition Bmal1 and KO at time t3 (12h).

- BmKo\_t4\_r2** The sample is the first replica (r2) of the biological condition Bmal1 and KO at time t4 (16h).
- BmKo\_t5\_r2** The sample is the first replica (r2) of the biological condition Bmal1 and KO at time t5 (20h).
- BmKo\_t0\_r3** The sample is the first replica (r3) of the biological condition Bmal1 and KO at time t0 (00h).
- BmKo\_t1\_r3** The sample is the first replica (r3) of the biological condition Bmal1 and KO at time t1 (04h).
- BmKo\_t2\_r3** The sample is the first replica (r3) of the biological condition Bmal1 and KO at time t2 (08h).
- BmKo\_t3\_r3** The sample is the first replica (r3) of the biological condition Bmal1 and KO at time t3 (12h).
- BmKo\_t4\_r3** The sample is the first replica (r3) of the biological condition Bmal1 and KO at time t4 (16h).
- BmKo\_t5\_r3** The sample is the first replica (r3) of the biological condition Bmal1 and KO at time t5 (20h).
- BmKo\_t0\_r4** The sample is the first replica (r4) of the biological condition Bmal1 and KO at time t0 (00h).
- BmKo\_t1\_r4** The sample is the first replica (r4) of the biological condition Bmal1 and KO at time t1 (04h).
- BmKo\_t2\_r4** The sample is the first replica (r4) of the biological condition Bmal1 and KO at time t2 (08h).
- BmKo\_t3\_r4** The sample is the first replica (r4) of the biological condition Bmal1 and KO at time t3 (12h).
- BmKo\_t4\_r4** The sample is the first replica (r4) of the biological condition Bmal1 and KO at time t4 (16h).
- BmKo\_t5\_r4** The sample is the first replica (r4) of the biological condition Bmal1 and KO at time t5 (20h).
- BmWt\_t0\_r5** The sample is the first replica (r5) of the biological condition Bmal1 and wild-type at time t0 (00h).
- BmWt\_t1\_r5** The sample is the first replica (r5) of the biological condition Bmal1 and wild-type at time t1 (04h).
- BmWt\_t2\_r5** The sample is the first replica (r5) of the biological condition Bmal1 and wild-type at time t2 (08h).
- BmWt\_t3\_r5** The sample is the first replica (r5) of the biological condition Bmal1 and wild-type at time t3 (12h).
- BmWt\_t4\_r5** The sample is the first replica (r5) of the biological condition Bmal1 and wild-type at time t4 (16h).
- BmWt\_t5\_r5** The sample is the first replica (r5) of the biological condition Bmal1 and wild-type at time t5 (20h).
- BmWt\_t0\_r6** The sample is the first replica (r6) of the biological condition Bmal1 and wild-type at time t0 (00h).

- BmWt\_t1\_r6** The sample is the first replica (r6) of the biological condition Bmal1 and wild-type at time t1 (04h).
- BmWt\_t2\_r6** The sample is the first replica (r6) of the biological condition Bmal1 and wild-type at time t2 (08h).
- BmWt\_t3\_r6** The sample is the first replica (r6) of the biological condition Bmal1 and wild-type at time t3 (12h).
- BmWt\_t4\_r6** The sample is the first replica (r6) of the biological condition Bmal1 and wild-type at time t4 (16h).
- BmWt\_t5\_r6** The sample is the first replica (r6) of the biological condition Bmal1 and wild-type at time t5 (20h).
- BmWt\_t0\_r7** The sample is the first replica (r7) of the biological condition Bmal1 and wild-type at time t0 (00h).
- BmWt\_t1\_r7** The sample is the first replica (r7) of the biological condition Bmal1 and wild-type at time t1 (04h).
- BmWt\_t2\_r7** The sample is the first replica (r7) of the biological condition Bmal1 and wild-type at time t2 (08h).
- BmWt\_t3\_r7** The sample is the first replica (r7) of the biological condition Bmal1 and wild-type at time t3 (12h).
- BmWt\_t4\_r7** The sample is the first replica (r7) of the biological condition Bmal1 and wild-type at time t4 (16h).
- BmWt\_t5\_r7** The sample is the first replica (r7) of the biological condition Bmal1 and wild-type at time t5 (20h).
- BmWt\_t0\_r8** The sample is the first replica (r8) of the biological condition Bmal1 and wild-type at time t0 (00h).
- BmWt\_t1\_r8** The sample is the first replica (r8) of the biological condition Bmal1 and wild-type at time t1 (04h).
- BmWt\_t2\_r8** The sample is the first replica (r8) of the biological condition Bmal1 and wild-type at time t2 (08h).
- BmWt\_t3\_r8** The sample is the first replica (r8) of the biological condition Bmal1 and wild-type at time t3 (12h).
- BmWt\_t4\_r8** The sample is the first replica (r8) of the biological condition Bmal1 and wild-type at time t4 (16h).
- BmWt\_t5\_r8** The sample is the first replica (r8) of the biological condition Bmal1 and wild-type at time t5 (20h).
- CrKo\_t0\_r9** The sample is the first replica (r9) of the biological condition Cry1/2 and KO at time t0 (00h).
- CrKo\_t1\_r9** The sample is the first replica (r9) of the biological condition Cry1/2 and KO at time t1 (04h).
- CrKo\_t2\_r9** The sample is the first replica (r9) of the biological condition Cry1/2 and KO at time t2 (08h).
- CrKo\_t3\_r9** The sample is the first replica (r9) of the biological condition Cry1/2 and KO at time t3 (12h).

- CrKo\_t4\_r9** The sample is the first replica (r9) of the biological condition Cry1/2 and KO at time t4 (16h).
- CrKo\_t5\_r9** The sample is the first replica (r9) of the biological condition Cry1/2 and KO at time t5 (20h).
- CrKo\_t0\_r10** The sample is the first replica (r10) of the biological condition Cry1/2 and KO at time t0 (00h).
- CrKo\_t1\_r10** The sample is the first replica (r10) of the biological condition Cry1/2 and KO at time t1 (04h).
- CrKo\_t2\_r10** The sample is the first replica (r10) of the biological condition Cry1/2 and KO at time t2 (08h).
- CrKo\_t3\_r10** The sample is the first replica (r10) of the biological condition Cry1/2 and KO at time t3 (12h).
- CrKo\_t4\_r10** The sample is the first replica (r10) of the biological condition Cry1/2 and KO at time t4 (16h).
- CrKo\_t5\_r10** The sample is the first replica (r10) of the biological condition Cry1/2 and KO at time t5 (20h).
- CrKo\_t0\_r11** The sample is the first replica (r11) of the biological condition Cry1/2 and KO at time t0 (00h).
- CrKo\_t1\_r11** The sample is the first replica (r11) of the biological condition Cry1/2 and KO at time t1 (04h).
- CrKo\_t2\_r11** The sample is the first replica (r11) of the biological condition Cry1/2 and KO at time t2 (08h).
- CrKo\_t3\_r11** The sample is the first replica (r11) of the biological condition Cry1/2 and KO at time t3 (12h).
- CrKo\_t4\_r11** The sample is the first replica (r11) of the biological condition Cry1/2 and KO at time t4 (16h).
- CrKo\_t5\_r11** The sample is the first replica (r11) of the biological condition Cry1/2 and KO at time t5 (20h).
- CrKo\_t0\_r12** The sample is the first replica (r12) of the biological condition Cry1/2 and KO at time t0 (00h).
- CrKo\_t1\_r12** The sample is the first replica (r12) of the biological condition Cry1/2 and KO at time t1 (04h).
- CrKo\_t2\_r12** The sample is the first replica (r12) of the biological condition Cry1/2 and KO at time t2 (08h).
- CrKo\_t3\_r12** The sample is the first replica (r12) of the biological condition Cry1/2 and KO at time t3 (12h).
- CrKo\_t4\_r12** The sample is the first replica (r12) of the biological condition Cry1/2 and KO at time t4 (16h).
- CrKo\_t5\_r12** The sample is the first replica (r12) of the biological condition Cry1/2 and KO at time t5 (20h).
- CrWt\_t0\_r13** The sample is the first replica (r13) of the biological condition Cry1/2 and wild-type at time t0 (00h).

- CrWt\_t1\_r13** The sample is the first replica (r13) of the biological condition Cry1/2 and wild-type at time t1 (04h).
- CrWt\_t2\_r13** The sample is the first replica (r13) of the biological condition Cry1/2 and wild-type at time t2 (08h).
- CrWt\_t3\_r13** The sample is the first replica (r13) of the biological condition Cry1/2 and wild-type at time t3 (12h).
- CrWt\_t4\_r13** The sample is the first replica (r13) of the biological condition Cry1/2 and wild-type at time t4 (16h).
- CrWt\_t5\_r13** The sample is the first replica (r13) of the biological condition Cry1/2 and wild-type at time t5 (20h).
- CrWt\_t0\_r14** The sample is the first replica (r14) of the biological condition Cry1/2 and wild-type at time t0 (00h).
- CrWt\_t1\_r14** The sample is the first replica (r14) of the biological condition Cry1/2 and wild-type at time t1 (04h).
- CrWt\_t2\_r14** The sample is the first replica (r14) of the biological condition Cry1/2 and wild-type at time t2 (08h).
- CrWt\_t3\_r14** The sample is the first replica (r14) of the biological condition Cry1/2 and wild-type at time t3 (12h).
- CrWt\_t4\_r14** The sample is the first replica (r14) of the biological condition Cry1/2 and wild-type at time t4 (16h).
- CrWt\_t5\_r14** The sample is the first replica (r14) of the biological condition Cry1/2 and wild-type at time t5 (20h).
- CrWt\_t0\_r15** The sample is the first replica (r15) of the biological condition Cry1/2 and wild-type at time t0 (00h).
- CrWt\_t1\_r15** The sample is the first replica (r15) of the biological condition Cry1/2 and wild-type at time t1 (04h).
- CrWt\_t2\_r15** The sample is the first replica (r15) of the biological condition Cry1/2 and wild-type at time t2 (08h).
- CrWt\_t3\_r15** The sample is the first replica (r15) of the biological condition Cry1/2 and wild-type at time t3 (12h).
- CrWt\_t4\_r15** The sample is the first replica (r15) of the biological condition Cry1/2 and wild-type at time t4 (16h).
- CrWt\_t5\_r15** The sample is the first replica (r15) of the biological condition Cry1/2 and wild-type at time t5 (20h).
- CrWt\_t0\_r16** The sample is the first replica (r16) of the biological condition Cry1/2 and wild-type at time t0 (00h).
- CrWt\_t1\_r16** The sample is the first replica (r16) of the biological condition Cry1/2 and wild-type at time t1 (04h).
- CrWt\_t2\_r16** The sample is the first replica (r16) of the biological condition Cry1/2 and wild-type at time t2 (08h).
- CrWt\_t3\_r16** The sample is the first replica (r16) of the biological condition Cry1/2 and wild-type at time t3 (12h).

**CrWt\_t4\_r16** The sample is the first replica (r16) of the biological condition Cry1/2 and wild-type at time t4 (16h).

**CrWt\_t5\_r16** The sample is the first replica (r16) of the biological condition Cry1/2 and wild-type at time t5 (20h).

### Details

The data is used in order to describe our algorithm in the case where samples belong to different time points.

We kept 500 genes only in order to increase the speed for each example.

### Value

Mouse times series dataset with four biological conditions and with 6 time measurements.

### Source

This dataset comes from Gene Expression Omnibus (GEO) <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE135898>. The name of the samples was renamed in order to be used with our package.

### References

Weger BD, Gobet C, David FPA, Atger F et al. 'Systematic analysis of differential rhythmic liver gene expression mediated by the circadian clock and feeding rhythms'. Proc Natl Acad Sci USA 2021 Jan 19;118(3). PMID:33452134. GEO:GSE135898.

### Examples

```
data(RawCounts_Weger2021_MOUSEsub500)
```

---

Results\_DEanalysis\_sub500

*DE results of three dataset*

---

### Description

The list Results\_DEanalysis\_sub500 contains the results of `DEanalysisGlobal()` for each of the following raw counts : RawCounts\_Weger2021\_MOUSEsub500, RawCounts\_Leong2014\_FISSIONsub500wt and RawCounts\_Schleiss2021\_CLLsub500

### Usage

```
data(Results_DEanalysis_sub500)
```

### Format

A list of 3 SummarizedExperiment class object

**Details**

Each list in Results\_DEanalysis\_sub500 contains only the necessary outputs of `DEanalysisGlobal()`, needed for the functions: `DEplotVolcanoMA()`, `DEplotHeatmaps()`, `GSEAPreprocessing()`, and `GSEAQuickAnalysis()`, for each of the following raw counts : RawCounts\_Weger2021\_MOUSEsub500, RawCounts\_Leong2014\_FISSIONsub500wt and RawCounts\_Schleiss2021\_CLLsub500

**Value**

Results\_DEanalysis\_sub500 contains the outputs of `DEanalysisGlobal()` of: RawCounts\_Weger2021\_MOUSEsub500, RawCounts\_Leong2014\_FISSIONsub500wt and RawCounts\_Schleiss2021\_CLLsub500

**Examples**

```
data(Results_DEanalysis_sub500)
```

---

```
Transcript_HomoSapiens_Database  
Homo sapiens transcript database
```

---

**Description**

The database is a data.frame which contains transcript length of homo sapiens genes (40452 genes).

**Usage**

```
data(Transcript_HomoSapiens_Database)
```

**Format**

A data frame with 500 rows (genes) and 13 columns (samples). The column names are as follow

**symbol** ENSEMBL gene names.

**Median.length.RNA** The sample is the first replica (r1) of the biological condition N1wt and T1wt.

**Details**

The first column contains genes symbol of the homo sapiens organism and the second column contains the median of transcript length for each gene of the first column.

**Value**

Mouse dataset with four biological conditions.

**Source**

HGNC, ENSEMBL and NCBI database.

**Examples**

```
data(Transcript_HomoSapiens_Database)
```



# Index

- \* **datasets**
  - RawCounts\_Antoszewski2022\_MOUSEsub500, [74](#)
  - RawCounts\_Leong2014\_FISSIONsub500wt, [75](#)
  - RawCounts\_Schleiss2021\_CLLsub500, [78](#)
  - RawCounts\_Weger2021\_MOUSEsub500, [82](#)
  - Results\_DEanalysis\_sub500, [87](#)
  - Transcript\_HomoSapiens\_Database, [88](#)
- \* **internal**
  - MultiRNAflow-package, [3](#)
- BiocGenerics::estimateSizeFactors(), [7](#), [8](#)
- CharacterNumbers, [4](#)
- CharacterNumbers(), [5](#), [15](#)
- ColnamesToFactors, [5](#)
- ColnamesToFactors(), [4](#), [6](#), [66](#)
- ComplexHeatmap::Heatmap(), [41](#)
- DATAnormalization, [3](#), [6](#)
- DATAnormalization(), [8–14](#), [17](#), [18](#), [55](#), [57–59](#), [61](#), [64](#), [65](#), [67](#), [68](#), [70–72](#)
- DATAplotBoxplotSamples, [9](#)
- DATAplotBoxplotSamples(), [7](#), [10](#)
- DATAplotExpression1Gene, [11](#)
- DATAplotExpression1Gene(), [12](#), [14](#)
- DATAplotExpressionGenes, [12](#)
- DATAplotExpressionGenes(), [12](#)
- DATAprepSE, [14](#)
- DATAprepSE(), [6–9](#), [11](#), [17](#), [18](#), [56](#), [59](#), [61](#), [64](#), [67](#), [68](#), [72](#)
- DEanalysisGlobal, [3](#), [17](#)
- DEanalysisGlobal(), [17](#), [19](#), [20](#), [24](#), [25](#), [28](#), [32](#), [39](#), [40](#), [44](#), [45](#), [50–53](#), [87](#), [88](#)
- DEanalysisGroup, [22](#)
- DEanalysisGroup(), [19](#), [35](#), [42](#), [46](#), [47](#)
- DEanalysisSubData, [25](#)
- DEanalysisTime, [26](#)
- DEanalysisTime(), [19](#), [34](#), [43](#)
- DEanalysisTimeAndGroup, [29](#)
- DEanalysisTimeAndGroup(), [20](#), [33–35](#), [37](#), [42](#), [43](#), [48](#), [49](#)
- DEplotAlluvial, [32](#)
- DEplotAlluvial(), [20](#), [21](#), [28](#), [31](#), [32](#), [34](#), [49](#)
- DEplotBarplot, [34](#)
- DEplotBarplot(), [19](#), [24](#), [35](#), [47](#), [49](#)
- DEplotBarplotFacetGrid, [36](#)
- DEplotBarplotFacetGrid(), [21](#), [31](#), [49](#)
- DEplotBarplotTime, [38](#)
- DEplotBarplotTime(), [20](#), [28](#), [49](#)
- DEplotHeatmaps, [39](#)
- DEplotHeatmaps(), [88](#)
- DEplotVennBarplotGroup, [41](#)
- DEplotVennBarplotGroup(), [19](#), [21](#), [24](#), [31](#), [47](#), [49](#)
- DEplotVennBarplotTime, [43](#)
- DEplotVennBarplotTime(), [20](#), [21](#), [28](#), [31](#), [49](#)
- DEplotVolcanoMA, [44](#)
- DEplotVolcanoMA(), [88](#)
- DEresultGroup, [46](#)
- DEresultGroupPerTime, [48](#)
- DESeq2::DESeq(), [3](#), [18](#), [22](#), [23](#), [27](#), [29](#), [46](#), [48](#)
- DESeq2::DESeqDataSetFromMatrix(), [14](#), [17](#)
- DESeq2::rlog(), [7](#), [8](#)
- DESeq2::vst(), [7](#), [8](#)
- factoextra::fviz\_dend(), [57](#)
- FactoMineR::HCPC(), [3](#), [55](#), [57](#), [61](#), [62](#)
- FactoMineR::PCA(), [3](#), [56](#), [65](#), [68](#), [70–72](#)
- ggplot2, [34](#), [35](#)
- ggplot2::facet\_grid(), [37](#)
- ggplot2::geom\_bar(), [34](#), [37](#)

ggplot2::geom\_boxplot, [10](#)  
ggplot2::geom\_boxplot(), [10](#)  
ggplot2::geom\_errorbar(), [11](#), [13](#)  
ggplot2::geom\_jitter, [10](#)  
ggplot2::geom\_violin(), [11](#), [13](#)  
gprofiler2::gost(), [3](#), [53](#), [54](#)  
graphics::persp(), [56](#), [64](#), [68](#)  
GSEAPreprocessing, [3](#), [50](#)  
GSEAPreprocessing(), [88](#)  
GSEAQuickAnalysis, [3](#), [52](#)  
GSEAQuickAnalysis(), [88](#)  
  
HCPCanalysis, [3](#), [55](#)  
HCPCanalysis(), [72](#)  
  
Mfuzz::mfuzz(), [59](#), [61](#)  
Mfuzz::mfuzz.plot2(), [3](#), [59](#)  
MFUZZanalysis, [3](#), [58](#)  
MFUZZanalysis(), [6](#), [62](#)  
MFUZZclustersNumber, [60](#)  
MFUZZclustersNumber(), [6](#), [58](#), [59](#)  
MultiRNAflow (MultiRNAflow-package), [3](#)  
MultiRNAflow-package, [3](#)  
  
PCAanalysis, [3](#), [63](#)  
PCAanalysis(), [67](#), [69](#), [72](#)  
PCAgraphics, [67](#)  
PCAgraphics(), [57](#), [66](#)  
PCAPreprocessing, [70](#)  
PCAPreprocessing(), [6](#), [72](#)  
PCArealization, [71](#)  
PCArealization(), [57](#), [67](#), [69–72](#)  
plot3Drgl::plotrgl(), [56](#), [57](#), [65](#), [68](#), [69](#)  
  
RawCounts\_Antoszewski2022\_MOUSEsub500,  
[74](#)  
RawCounts\_Leong2014\_FISSIONsub500wt,  
[75](#)  
RawCounts\_Schleiss2021\_CLLsub500, [78](#)  
RawCounts\_Weger2021\_MOUSEsub500, [82](#)  
RawCountsSimulation, [73](#)  
Results\_DEanalysis\_sub500, [87](#)  
  
stats::kmeans(), [61](#), [62](#)  
stats::p.adjust(), [18](#), [23](#), [27](#), [29](#), [30](#), [46](#), [48](#)  
SummarizedExperiment::SummarizedExperiment(),  
[14](#), [17](#)  
  
Transcript\_HomoSapiens\_Database, [88](#)  
  
UpSetR::upset(), [42](#), [43](#)