Package: ribiosNGS (via r-universe)

May 16, 2026

Type Package

Title Differential gene expression analysis for next-generation sequencing data in the 'ribios' suite

Version 1.0.0

Description Provides a comprehensive workflow for differential gene expression analysis of RNA-seq data using 'edgeR' and 'limma'-voom pipelines. The package supports count filtering, normalization, surrogate variable analysis, batch correction, and visualization including volcano plots, smear plots, and PCA. It integrates with the 'ribios' suite of packages and is designed for reproducible bioinformatics analyses.

Depends R (>= 4.4.0),

Imports BiocGenerics, ribiosExpression (>= 1.2.0), edgeR, ribiosAnnotation (>= 3.6.0), ribiosUtils, methods, Biobase, lattice, ribiosIO (>= 1.0-60), ribiosPlot (>= 1.2-9), graphics, limma, readr, readxl, magrittr, sva, dplyr, vsn, made4, ggplot2, rlang, gt, scales, UpSetR, ggrepel, reshape2

Suggests testthat, ggpubr, tibble, knitr, rmarkdown, BiocStyle

biocViews DifferentialExpression, GeneExpression, RNASeq, Transcriptomics

License GPL-3

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.3

Collate 'utils.R' 'ribiosNGS-package.R' 'AllClasses.R' 'AllGenerics.R' 'AllMethods.R' '20210217-toribios.R' 'DGEListToLongTable.R' 'EdgeObject.R' 'annotateMethods.R' 'pseudoT.R' 'appendZScore.R' 'edgeR-funcs.R' 'dge.R' 'dgeTable.R' 'edgeR-slurm-lsf.R' 'eset2DGEList.R' 'exportEdgeResult.R' 'filterByCPM.R' 'knockdown.R' 'logFCmatrix.R' 'logFCpca.R' 'mergeDGEList.R' 'minGroupCount.R' 'mps-io.R' 'normalize.R' 'pca.R' 'plotMethods.R' 'plotTopSigGenes.R' 'readBiokitAsDGEList.R'

'readFeatureAnnotationForEdgeR.R'
'readSampleAnnotationForEdgeR.R' 'renameContrast.R'
'split.DGEList.R' 'staticGeneLevelPlots.R' 'sva.R'
'svaseqRemove.R' 'topDgeExpression.R' 'tpm.R' 'voomLimma.R'
'writeBiokitSampleAnnotation.R' 'writeDGEList.R'

Remotes github::bedapub/ribiosExpression,
github::bedapub/ribiosAnnotation, github::bedapub/ribiosUtils,
github::bedapub/ribiosIO, github::bedapub/ribiosPlot

Additional_repositories https://bedapub.r-universe.dev

URL https://github.com/bedapub/ribiosNGS

BugReports https://github.com/bedapub/ribiosNGS/issues

Config/pak/sysreqs cmake make libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev perl libsasl2-dev libnode-dev libx11-dev zlib1g-dev

Repository https://bedapub.r-universe.dev

Date/Publication 2026-02-15 11:40:09 UTC

RemoteUrl https://github.com/bedapub/ribiosNGS

RemoteRef HEAD

RemoteSha 90513e37c7ab00a23dff38258a29b2e7e4589443

Contents

`annotate,EdgeObject,character,logical-method`
Annotate an EdgeObject

Description

Annotate an EdgeObject

Usage

```
## S4 method for signature 'EdgeObject,character,logical'
annotate(object, target, check.target)
```

Arguments

`object`	An EdgeObject.
`target`	Character, target of annotation.
`check.target`	Logical, check whether the target is valid or not.

Value

An annotated `EdgeObject`.

`annotate,EdgeObject,character,missing-method`
Annotate an EdgeObject, without checking the target

Description

Annotate an EdgeObject, without checking the target

Usage

```
## S4 method for signature 'EdgeObject,character,missing'
annotate(object, target)
```

Arguments

`object`	An EdgeObject
`target`	Character, target of annotation

Value

An annotated `EdgeObject`.

`annotate,EdgeObject,missing,missing-method`
Annotate an EdgeObject automatically without checking the target

---

Description

Annotate an EdgeObject automatically without checking the target

Usage

```
## S4 method for signature 'EdgeObject,missing,missing'
annotate(object)
```

Arguments

`object`	An EdgeObject

Value

An annotated `EdgeObject`.

annotation,EdgeObject-method *Get annotation information from an EdgeObject*

Description

Get annotation information from an EdgeObject

Usage

```
## S4 method for signature 'EdgeObject'
annotation(object)
```

Arguments

object	An EdgeObject.

Value

A character string, indicating the annotation type

appendPseudoT *Append degree of freedom and pseudo t-statistics to dgeTable*

Description

Append degree of freedom and pseudo t-statistics to dgeTable

Usage

```
appendPseudoT(edgeResult, dgeTable)
```

Arguments

edgeResult	An EdgeResult object
dgeTable	A data.frame, derived from dgeTables or dgeTable usually.

Value

A new data.frame with two new columns, df and pseudoT, containing degree of freem and the (pseudo) t-statistic, respective.y The function relies on the fact that the degree of freedom (‘df.residual‘ in GLM result) is the same for all genes. If this is not the case, it will use the ‘FeatureName‘ column in gene annotation to match the degree of freedoms. The function is only used internally by dgeTablesWithPseudoT and dgeTableWithPseudoT.

Note

the lower.tail option in qt is not vectorized, therefore the sign should be provided separately.

appendRanks *Append ranks to dgeTbl*

Description

Append ranks to dgeTbl

Usage

```
appendRanks(dgeTbl)
```

Arguments

dgeTbl	A dgeTbl, a data.frame that at least contain following columns logFC and PValue

Value

The dgeTble with four columns appended: rank_logFC, rank_PValue, rank_absLogFC, and total_features, and sorted by increasing P-values

Examples

```
myTbl <- data.frame(GeneSymbol=LETTERS[1:5],
  logFC=rnorm(5), PValue=runif(5, 0, 1))
appendRanks(myTbl)
```

appendZScore *Append zscores to dgeTable*

Description

Append zscores to dgeTable

Usage

```
appendZScore(dgeTable)
```

Arguments

dgeTable	A data.frame, derived from dgeTables or dgeTable usually. It must contain the following two columns: PValue, and logFC (case sensitive).

Value

A new `data.frame` with one new column, `zScore`, containing the z-score transformed from p-values. If that column exists, it will be rewritten. The function is similar to `appendPseudoT`, with the difference that the Gaussian distribution is used instead of the t-distribution. This solution delivers less extreme values, because t-distribution is heavy-tailed. It allows comparison between studies of different sample sizes.

See Also

`appendPseudoT`

---

assertEdgeToptable *Assert that the input data.frame is a valid EdgeTopTable*

Description

Assert that the input data.frame is a valid EdgeTopTable

Usage

```
assertEdgeToptable(x)
```

Arguments

x	A data.frame

Value

Logical

---

aveExpr *Get aveExpr threshold in LimmaSigFilter*

Description

Get aveExpr threshold in LimmaSigFilter

Usage

```
aveExpr(limmaSigFilter)
```

Arguments

`limmaSigFilter`	An `EdgeSigFilter` object

Value

Numeric values of the logCPM filter

BCV *Return a data.frame of BCV values*

Description

Return a data.frame of BCV values

Usage

```
BCV(x)

## S4 method for signature 'DGEList'
BCV(x)

## S4 method for signature 'EdgeResult'
BCV(x)
```

Arguments

x	An object

Value

A `data.frame` of BCV values.

Methods (by class)

- `BCV(DGEList)`: Method for DGEList
- `BCV(EdgeResult)`: Method for EdgeResult

`boxplot,EdgeObject-method` *Boxplot of an EdgeObject*

Description

Boxplot of an EdgeObject

Usage

```
## S4 method for signature 'EdgeObject'
boxplot(
  x,
  type = c("normFactors", "modLogCPM"),
  xlab = "",
  ylab = NULL,
  main = "",
  ...
)
```

Arguments

x	An EdgeObject
type	The type of boxplot: 'normFactors' and 'modLogCPM' are supported.
xlab	Character, xlab.
ylab	Character, ylab.
main	Character, title.
...	Passed to boxplot.

Value

Called for its side effect of plotting; returns invisibly NULL.

---

calcNormFactorsIfNot *Calculate normalisation factor if not*

Description

Calculate normalisation factor if not

Usage

```
calcNormFactorsIfNot(dgeList)
```

Arguments

dgeList	A DGEList object Calculate the normalisation factors if not done yet

Value

Updated dgeList object with norm.factors filled

checkBiokitSampleAnnotation *Check sample annotation data.frame or tibble meets the requirement of the biokit pipeline*

Description

Check sample annotation data.frame or tibble meets the requirement of the biokit pipeline

Usage

```
checkBiokitSampleAnnotation(df)
```

Arguments

df	Sample annotation, can be either a `data.frame` or `tbl_df` object

Value

Invisible `NULL` if the requirement is met, otherwise an error is printed and the function stops

The biokit pipeline requires that values in each column contain no empty spaces. This function ensures that.

See Also

`writeBiokitSampleAnnotation`, which calls this function to ensure that the sample annotation file is ok

Examples

```
test <- data.frame(Char=LETTERS[1:6],
                   Integer=1:6,
                   Number=pi*1:6,
                Factor=gl(2,3, labels = c("level 1", "level 2")), stringsAsFactors=FALSE)
testthat::expect_error(checkBiokitSampleAnnotation(test),
                       regexp = "level 1.*level 2")
testFix <- data.frame(Char=LETTERS[1:6],
                      Integer=1:6,
                      Number=pi*1:6,
                  Factor=gl(2,3, labels = c("level1", "level2")), stringsAsFactors=FALSE)
testthat::expect_silent(checkBiokitSampleAnnotation(testFix))
if(require("tibble")) {
  testthat::expect_error(checkBiokitSampleAnnotation(tibble::as_tibble(test)),
                         regexp = "level 1.*level 2")
  testthat::expect_silent(checkBiokitSampleAnnotation(tibble::as_tibble(testFix)))
}
```

checkContrastNames *Check a contrast matrix to make sure that it is likely o.k.*

Description

Check a contrast matrix to make sure that it is likely o.k.

Usage

```
checkContrastNames(contrastMatrix, action = c("message", "warning", "error"))
```

Arguments

contrastMatrix	A contrast matrix
action	Character strings, the action to perform in case the names show irregularities Right now, the function checks no column names contain the equal sign.

Value

Invisibly returns NULL. Called for its side effect of issuing a message, warning, or error if column names contain equal signs.

Examples

```
testDesign <- cbind(Control=rep(1,8), Treatment=rep(c(0,1),4), Batch=rep(c(0, 1), each=4))
problemContrast <- limma::makeContrasts("Treatment"="Treatment",
  "Batch=Batch", ## problematic
  levels=testDesign)
checkContrastNames(problemContrast, action="message")
if(requireNamespace("testthat")) {
  testthat::expect_warning(checkContrastNames(problemContrast,
         action="warning"))
  testthat::expect_error(checkContrastNames(problemContrast,
         action="error"))
}
```

commonBCV *Common biological coefficients of variance (BCV)*

Description

Common biological coefficients of variance (BCV)

Usage

```
commonBCV(x)

## S4 method for signature 'DGEList'
commonBCV(x)

## S4 method for signature 'EdgeResult'
commonBCV(x)
```

Arguments

x	An object

Value

A numeric value of the common BCV.

Methods (by class)

- `commonBCV(DGEList)`: method for DGEList
- `commonBCV(EdgeResult)`: method for EdgeResult

---

commonDisp *Common dispersion*

Description

Common dispersion

Usage

```
commonDisp(object)

## S4 method for signature 'DGEList'
commonDisp(object)

## S4 method for signature 'EdgeObject'
commonDisp(object)
```

Arguments

object	An object

Value

A numeric value of the common dispersion.

Methods (by class)

- `commonDisp(DGEList)`: Method for DGEList
- `commonDisp(EdgeObject)`: Method for EdgeObject

`commonDisp<-` *commonDisp-set*

Description

Set common dispersion

Usage

```
commonDisp(object) <- value

## S4 replacement method for signature 'DGEList,numeric'
commonDisp(object) <- value

## S4 replacement method for signature 'EdgeObject,numeric'
commonDisp(object) <- value
```

Arguments

`object`	An object
`value`	Numeric value

Value

The updated object with the common dispersion set.

Functions

- `commonDisp(object = DGEList) <- value`: Method for DGEList
- `commonDisp(object = EdgeObject) <- value`: Method for EdgeObject

contrastMatrix-set *Assign contrast matrix*

Description

Assign contrast matrix

Usage

```
contrastMatrix(object) <- value

## S4 replacement method for signature 'EdgeObject,matrix'
contrastMatrix(object) <- value
```

Arguments

object	An object
value	Matrix

Value

The updated object with the new contrast matrix.

Functions

- `contrastMatrix(object = EdgeObject) <- value`: Method for EdgeObject

contrastMatrix,EdgeObject-method *Extract contrast matrix from an EdgeObject object*

Description

Extract contrast matrix from an EdgeObject object

Usage

```
## S4 method for signature 'EdgeObject'
contrastMatrix(object)
```

Arguments

object	An EdgeObject object

Value

A contrast matrix.

`contrastMatrix,EdgeResult-method`
Extract contrast matrix from an EdgeResult object

Description

Extract contrast matrix from an EdgeResult object

Usage

```
## S4 method for signature 'EdgeResult'
contrastMatrix(object)
```

Arguments

`object`	An EdgeResult object

Value

A contrast matrix.

`contrastNames,EdgeObject-method`
Extract contrast names from an EdgeObject object

Description

Extract contrast names from an EdgeObject object

Usage

```
## S4 method for signature 'EdgeObject'
contrastNames(object)
```

Arguments

`object`	An EdgeObject object

Value

A character vector of contrast names.

contrastNames,EdgeResult-method

Extract contrast names from an EdgeResult object

Description

Extract contrast names from an EdgeResult object

Usage

```
## S4 method for signature 'EdgeResult'
contrastNames(object)
```

Arguments

object	An EdgeResult object

Value

A character vector of contrast names.

contrastSampleIndices,EdgeResult,character-method

Extract contrast sample indices

Description

Extract contrast sample indices

Usage

```
## S4 method for signature 'EdgeResult,character'
contrastSampleIndices(object, contrast)
```

Arguments

object	An EdgeResult object.
contrast	Character, indicating the contrast of interest.

Value

A list of integer vectors indicating sample indices for each group in the contrast.

contrastSampleIndices,EdgeResult,integer-method
Extract contrast sample indices

Description

Extract contrast sample indices

Usage

```
## S4 method for signature 'EdgeResult,integer'
contrastSampleIndices(object, contrast)
```

Arguments

object	An EdgeResult object.
contrast	Character, indicating the contrast of interest.

Value

A list of integer vectors indicating sample indices for each group in the contrast.

countByGroup *Sample counts by group*

Description

Sample counts by group

Usage

```
countByGroup(edgeObj)

maxCountByGroup(edgeObj)

hasNoReplicate(edgeObj)
```

Arguments

edgeObj	An EdgeObject object

Value

A named vector if integers, sample counts by group

Functions

- `maxCountByGroup()`: Returns the max count
- `hasNoReplicate()`: Returns `TRUE` if the largest group has only one sample

`CountDgeResult-class` *Object that contains count data, dgeTables, and sigFilter*

Description

Object that contains count data, dgeTables, and sigFilter

Slots

`dgeTables` A list of dgeTable

`sigFilter` Significantly regulated gene filter

Note

The object is used only for inheritance

`counts,DGEList-method` *Return counts in a DGEList object*

Description

Return counts in a DGEList object

Usage

```
## S4 method for signature 'DGEList'
counts(object)
```

Arguments

`object`	A `DGEList` object.

Value

A numeric matrix of counts.

counts,EdgeObject-method *Return counts in EdgeObject*

Description

Return counts in EdgeObject

Usage

```
## S4 method for signature 'EdgeObject'
counts(object, filter = TRUE)
```

Arguments

object	An EdgeObject
filter	Logical, whether filtered matrix (by default) or unfiltered matrix should be returned

Value

A numeric matrix of counts.

See Also

filterByCPM

countsRemoveSV *Apply SVA to transformed count data and return the transformed matrix removing the effect of surrogate variables*

Description

Apply SVA to transformed count data and return the transformed matrix removing the effect of surrogate variables

Usage

```
countsRemoveSV(
  counts,
  designMatrix,
  transformFunc = function(counts, designMatrix) voom(counts, designMatrix)$E
)
```

Arguments

counts	A matrix of counts
designMatrix	Design matrix
transformFunc	A function to transform the count data

Value

The expression matrix, with SV effects removed

Examples

```
exCounts <- matrix(rpois(12000, 10), nrow=2000, ncol=6)
exCounts[1:100, 2:3] <- exCounts[1:100,2:3]+20
exDesign <- model.matrix(~gl(2,3))
head(countsRemoveSV(exCounts, designMatrix=exDesign))
```

---

countsSVA *Apply SVA to transformed count data*

Description

Apply SVA to transformed count data

Usage

```
countsSVA(
  counts,
  designMatrix,
  transformFunc = function(counts, designMatrix) voom(counts, designMatrix)$E,
  ...
)
```

Arguments

counts	A matrix of counts
designMatrix	Design matrix
transformFunc	A function to transform the count data
...	Passed to inferSV

Value

The SV matrix

Examples

```
set.seed(1887)
exCounts <- matrix(rpois(12000, 10), nrow=2000, ncol=6)
exCounts[1:100, 2:3] <- exCounts[1:100,2:3]+20
exDesign <- model.matrix(~gl(2,3))
countsSVA(exCounts, designMatrix=exDesign)
```

---

cpm.EdgeObject *cpm for EdgeObject*

Description

cpm for EdgeObject

Usage

```
## S3 method for class 'EdgeObject'
cpm(y, ...)
```

Arguments

y	An EdgeObject object
...	Passed to cpm

Value

A numeric matrix of counts per million values.

See Also

cpm

---

cpmFilter *Filter by counts per million (cpm)*

Description

Filter by counts per million (cpm)

Usage

```
cpmFilter(object)
```

Arguments

object	An object

Value

The filtered object.

`cpmRemoveSV` *Apply cpm to voom-transformed count data, and return the voom expression matrix with surrogate variables' effect removed*

Description

Apply cpm to voom-transformed count data, and return the voom expression matrix with surrogate variables' effect removed

Usage

```
cpmRemoveSV(counts, designMatrix)
```

Arguments

counts	A matrix of counts
designMatrix	Design matrix

Value

The cpm matrix, with SV effects removed

Examples

```
exCounts <- matrix(rpois(12000, 10), nrow=2000, ncol=6)
exCounts[1:100, 2:3] <- exCounts[1:100,2:3]+20
exDesign <- model.matrix(~gl(2,3))
head(cpmRemoveSV(exCounts, designMatrix=exDesign))
## compare the results without SV removal, note the values in the second
## and third column are much larger than the rest
head(cpm(exCounts))
```

cpmSVA *Apply SVA to cpm-transformed count data*

Description

Apply SVA to cpm-transformed count data

Usage

```
cpmSVA(counts, designMatrix)
```

Arguments

counts	A matrix of counts
designMatrix	Design matrix

Value

The SV matrix

Examples

```
exCounts <- matrix(rpois(12000, 10), nrow=2000, ncol=6)
exCounts[1:100, 2:3] <- exCounts[1:100,2:3]+20
exDesign <- model.matrix(~gl(2,3))
cpmSVA(exCounts, designMatrix=exDesign)
```

customSmearPlot *Custom smear plot*

Description

Custom smear plot

Usage

```
customSmearPlot(
  tbl,
  main,
  xlab,
  ylab,
  pch = 19,
  cex = 0.2,
  smearWidth = 0.5,
  panel.first = grid(),
```

```
  smooth.scatter = FALSE,
  lowess = FALSE,
  ...
)
```

Arguments

tbl	A data.frame
main	Character string, title of the plot
xlab	Character string, xlab
ylab	Character string, ylab
pch	Point symbol
cex	Font size
smearWidth	Smear with
panel.first	Passed to maPlot.
smooth.scatter	Passed to maPlot.
lowess	Passed to maPlot.
...	Passed to maPlot.

Value

Called for its side effect of plotting; returns invisibly NULL.

---

designContrast *Retrieve the design/contrast object*

Description

Retrieve the design/contrast object

Usage

```
designContrast(edgeObject)
```

Arguments

edgeObject	An EdgeObject

Value

A DesignContrast object.

Examples

```
designContrast(exampleEdgeObject())
```

designMatrix-set *Assign the design matrix*

Description

Assign the design matrix

Usage

```
designMatrix(object) <- value

## S4 replacement method for signature 'EdgeObject,matrix'
designMatrix(object) <- value
```

Arguments

object	An object
value	Matrix

Value

The updated object with the new design matrix.

Functions

- designMatrix(object = EdgeObject) <- value: Method for EdgeObject

designMatrix,EdgeObject-method *Extract design matrix from an EdgeObject object*

Description

Extract design matrix from an EdgeObject object

Usage

```
## S4 method for signature 'EdgeObject'
designMatrix(object)
```

Arguments

object	An EdgeObject object

Value

A design matrix.

designMatrix,EdgeResult-method *Extract design matrix from an EdgeResult object*

Description

Extract design matrix from an EdgeResult object

Usage

```
## S4 method for signature 'EdgeResult'
designMatrix(object)
```

Arguments

object	An EdgeResult object

Value

A design matrix.

dgeGML *Return the dgeGML method*

Description

Return the dgeGML method

Usage

```
dgeGML(edgeResult)
```

Arguments

edgeResult	An EdgeResult object.

Value

A DGEGLM object.

dgeList *Extract DGEList from an EdgeObject*

Description

Extract DGEList from an EdgeObject

Extract DGEList from an EdgeResult object

Usage

```
dgeList(object)

## S4 method for signature 'EdgeObject'
dgeList(object)

## S4 method for signature 'EdgeResult'
dgeList(object)
```

Arguments

object	An EdgeResult object

Value

A DGEList object.

Methods (by class)

- dgeList(EdgeObject): Extract DGEList from EdgeObject
- dgeList(EdgeResult): Extract DGEList from EdgeResult

DGEListList *Construct a DGEListList object*

Description

Construct a DGEListList object

Usage

```
DGEListList(...)
```

Arguments

...	A list of DGEListList objects, can be passed as individual objects or in a list

Value

A DGEListList object.

DGEListList-class *An S4 class to represent a list of DGEListList objects*

Description

An S4 class to represent a list of DGEListList objects

DGEListToLongTable *Convert a DGEList object to a long data.frame containing expression, feature annotation, and sample annotation*

Description

Convert a DGEList object to a long data.frame containing expression, feature annotation, and sample annotation

Usage

```
DGEListToLongTable(x, exprsFun = function(dgeList) cpm(dgeList, log = TRUE))
```

Arguments

x	A DGEList object
exprsFun	A function to convert counts to expression data. Default: logCPM

Value

A long-format data.frame with expression values and annotations.

Note

Columns with empty names will be discard.

Examples

```
mat <- matrix(rnbinom(100, mu=5, size=2), ncol=10)
rownames(mat) <- sprintf("gene%d", 1:nrow(mat))
y <- edgeR::DGEList(counts=mat, group=rep(1:2, each=5))
DGEListToLongTable(y)
```

dgeTable *Return the top table in a unified format*

Description

Return the top table in a unified format

Usage

```
dgeTable(edgeResult, contrast = NULL)
```

Arguments

edgeResult	An EdgeResult object
contrast	Character, contrast name of interest. If NULL, all tables are returned in a rbind-form.

Value

A data.frame

dgeTableList *Return the top tables of specified contrast(s) in a list*

Description

Return the top tables of specified contrast(s) in a list

Usage

```
dgeTableList(edgeResult, contrast = NULL)
```

Arguments

edgeResult	An EdgeResult object
contrast	Character, contrast name(s) of interest. If NULL, all tables are returned in a rbind-form.

Value

A list of data.frames

dgeTables *Return a list of differential gene expression tables*

Description

Return a list of differential gene expression tables

Usage

```
dgeTables(edgeResult)
```

Arguments

edgeResult	An EdgeResult object

Value

A list of data.frames, each containing the DGEtable for one contrast.

See Also

dgeTable which returns one data.frame for one or more given contrasts.

dgeTablesWithPseudoT *Append dgeTables with pseudo t-statistic*

Description

Append dgeTables with pseudo t-statistic

Usage

```
dgeTablesWithPseudoT(edgeResult)
```

Arguments

edgeResult	An EdgeResult object.

Value

Similar as dgeTables, a list of data.frame, but with additional columns df (degree of freedom) and pseudoT. The pseudo t-statistic is calculated based on the P-value of the likelihood ratio test and the residual degree of freedom by the function. qt, and its sign is given by the sign of logFC.

See Also

dgeTableWithPseudoT

dgeTablesWithZScore *Append dgeTables with z-scores*

Description

Append dgeTables with z-scores

Usage

```
dgeTablesWithZScore(edgeResult)
```

Arguments

edgeResult	An EdgeResult object.

Value

Similar as dgeTables, a list of data.frame, but with an additional column zScore.

See Also

appendZScore, dgeTableWithZScore, dgeTableWithPseudoT

dgeTableWithPseudoT *Append dgeTable with pseudo t-statistic*

Description

Append dgeTable with pseudo t-statistic

Usage

```
dgeTableWithPseudoT(edgeResult, contrast = NULL)
```

Arguments

edgeResult	An EdgeResult object.
contrast	A character string, or integer index, or NULL, to specify the contrast. If NULL, results of all contrasts are returned.

Value

Similar as dgeTable, a data.frame, but with additional columns df (degree of freedom) and pseudoT. The pseudo t-statistic is calculated based on the P-value of the likelihood ratio test and the residual degree of freedom by the function qt, and its sign is given by the sign of logFC.

See Also

dgeTablesWithPseudoT, dgeTable

---

dgeTableWithZScore *Append dgeTable with z-scores*

Description

Append dgeTable with z-scores

Usage

```
dgeTableWithZScore(edgeResult, contrast = NULL)
```

Arguments

edgeResult	An EdgeResult object.
contrast	A character string, or integer index, or NULL, to specify the contrast. If NULL, results of all contrasts are returned.

Value

Similar as dgeTable, a data.frame, with an additional column zScore.

See Also

dgeTablesWithZScore, dgeTable, dgeTableWithPseudoT

---

dgeWithEdgeR *Perform differential gene expression analysis with edgeR*

Description

Perform differential gene expression analysis with edgeR

Usage

```
dgeWithEdgeR(edgeObj)
```

Arguments

edgeObj	An object of EdgeObject The function performs end-to-end differential gene expression (DGE) analysis with common best practice using edgeR

Value

An EdgeResult object

Examples

```
exMat <- matrix(rpois(120, 10), nrow=20, ncol=6)
exGroups <- gl(2,3, labels=c("Group1", "Group2"))
exDesign <- model.matrix(~0+exGroups)
colnames(exDesign) <- levels(exGroups)
exContrast <- matrix(c(-1,1), ncol=1, dimnames=list(c("Group1", "Group2"), c("Group2.vs.Group1")))
exDescon <- DesignContrast(exDesign, exContrast, groups=exGroups)
exFdata <- data.frame(Identifier=sprintf("Gene%d", 1:nrow(exMat)))
exPdata <- data.frame(Name=sprintf("Sample%d", 1:ncol(exMat)),
                      Group=exGroups)
exObj <- EdgeObject(exMat, exDescon,
                      fData=exFdata, pData=exPdata)
exDgeRes <- dgeWithEdgeR(exObj)
dgeTable(exDgeRes)
```

---

dgeWithLimmaVoom *Perform differential gene expression analysis with edgeR-limma*

Description

Perform differential gene expression analysis with edgeR-limma

Usage

```
dgeWithLimmaVoom(edgeObj, ...)
```

Arguments

edgeObj	An object of EdgeObject
...	Passed to voom
	The function performs end-to-end differential gene expression (DGE) analysis with common best practice using voom-limma

Value

A LimmaVoomResult object.

Examples

```
set.seed(1887)
exObj <- exampleEdgeObject()
exLimmaVoomRes <- dgeWithLimmaVoom(exObj)
dgeTable(exLimmaVoomRes)

## compare with edgeR
dgeTable(dgeWithEdgeR(exObj))

## LimmaVoomResult can be also used with exportEdgeResult
exportEdgeResult(exLimmaVoomRes, tempdir(), "overwrite")
```

---

dim.EdgeResult *Dimensions of an EdgeResults*

Description

Dimensions of an EdgeResults

Usage

```
## S3 method for class 'EdgeResult'
dim(x)
```

Arguments

x	An EdgeResult object

Value

An integer vector of length two (features, samples).

---

dispGroups,EdgeObject-method *Get display labels of sample groups*

Description

Get display labels of sample groups

Usage

```
## S4 method for signature 'EdgeObject'
dispGroups(object)
```

Arguments

object	An EdgeObject object

Value

A character vector of display labels for sample groups.

---

doSVA *Perform surrogate variable analysis (SVA) to an EdgeObject object*

Description

Perform surrogate variable analysis (SVA) to an EdgeObject object

Usage

```
doSVA(edgeObj, transform = c("voom", "cpm"))
```

Arguments

edgeObj	An EdgeObject object
transform	Function name to perform transformation, currently supported values include voom and cpm The count data associated with the EdgeObject object is first transformed, and surrogate variables are estimated from the transformed data. Correspondingly the design matrix and contrast matrix associated with the object are updated, too.

Value

An updated EdgeObject with the design and contrast matrices augmented by surrogate variables, if any are detected.

Examples

```
set.seed(1887)
exMat <- matrix(rpois(12000, 10), nrow=2000, ncol=6)
exMat[1:100,2:3] <- exMat[1:100, 2:3]+20
exGroups <- gl(2,3, labels=c("Group1", "Group2"))
exDesign <- model.matrix(~exGroups)
exContrast <- matrix(c(-1,1), ncol=1,
             dimnames=list(c("Group1", "Group2"), c("Group2.vs.Group1")))
exDescon <- DesignContrast(exDesign, exContrast, groups=exGroups)
exFdata <- data.frame(GeneSymbol=sprintf("Gene%d", 1:nrow(exMat)))
exPdata <- data.frame(Name=sprintf("Sample%d", 1:ncol(exMat)),
                      Group=exGroups)
exObj <- EdgeObject(exMat, exDescon,
                      fData=exFdata, pData=exPdata)
```

```
exSVAobj <- doSVA(exObj, transform="voom")
designMatrix(exSVAobj)
contrastMatrix(exSVAobj)

## Note that the SVA is sensitive against parameterisation, see
## the example below. Also notice that in the zero-intercept parameterisation,
## the SVA does not give meaningful results.
designMatrix(exObj) <- model.matrix(~0+exGroups)
designMatrix(doSVA(exObj, transform="voom"))
```

`EdgeObject` *Construct an EdgeObject object by a count matrix and DesignContrast*

Description

Construct an EdgeObject object by a count matrix and DesignContrast

Usage

```
EdgeObject(object, designContrast, ...)
```

Arguments

`object`	A matrix containing counts of features
`designContrast`	A `DesignContrast` object
`...`	Other parameters

Value

An `EdgeObject`.

Examples

```
exMat <- matrix(rpois(120, 10), nrow=20, ncol=6)
exGroups <- gl(2,3, labels=c("Group1", "Group2"))
exDesign <- model.matrix(~0+exGroups)
exContrast <- matrix(c(-1,1), ncol=1,
    dimnames=list(c("Group1", "Group2"), c("Group2.vs.Group1")))
exDescon <- DesignContrast(exDesign, exContrast, groups=exGroups)
exFdata <- data.frame(Identifier=sprintf("Gene%d", 1:nrow(exMat)))
exPdata <- data.frame(Name=sprintf("Sample%d", 1:ncol(exMat)),
                      Group=exGroups)
exObj <- EdgeObject(exMat, exDescon)
exObj2 <- EdgeObject(exMat, exDescon, fData=exFdata)
exObj3 <- EdgeObject(exMat, exDescon,
                      fData=exFdata, pData=exPdata)
```

```
fData(exObj3)

dgeList <- edgeR::DGEList(counts=exMat, samples=exPdata, genes=exFdata)
exObj4 <- EdgeObject(dgeList, exDescon)

## note that pData are appended after count information
pData(exObj2)
pData(exObj3)
```

`EdgeObject-class` *EdgeObject argumenting DGEList by including designContrast information*

Description

EdgeObject argumenting DGEList by including designContrast information

Usage

```
## S4 method for signature 'matrix,DesignContrast'
EdgeObject(
  object,
  designContrast,
  fData = NULL,
  pData = NULL,
  remove.zeros = FALSE
)

## S4 method for signature 'FeatAnnoExprs,DesignContrast'
EdgeObject(object, designContrast, pData = NULL, remove.zeros = FALSE)

## S4 method for signature 'DGEList,DesignContrast'
EdgeObject(object, designContrast)
```

Arguments

`object`	A `DGEList` object
`designContrast`	A `DesignContrast` object
`fData`	A `data.frame` containing annotation information for genes
`pData`	A `data.frame` containing annotation information for samples
`remove.zeros`	Logical, whether to remove rows that have 0 total count

Methods (by generic)

- `EdgeObject(object = matrix, designContrast = DesignContrast)`: The method for matrix as input

- EdgeObject(object = FeatAnnoExprs, designContrast = DesignContrast): The method for FeatAnnoExprs as input
- EdgeObject(object = DGEList, designContrast = DesignContrast): The method for DGEList as input

Slots

dgeList A DGEList object

designContrast A designContrast object

---

edgeRcommand *Export an DGEList, designMatrix, and contrastMatrix to files and return the command to run the edgeR script*

Description

Export an DGEList, designMatrix, and contrastMatrix to files and return the command to run the edgeR script

Usage

```
edgeRcommand(
  dgeList,
  designMatrix,
  contrastMatrix,
  outdir = "edgeR_output",
  outfilePrefix = "an-unnamed-project-",
  mps = FALSE,
  limmaVoom = FALSE,
  appendGmt = NULL,
  debug = FALSE,
  rootPath = "/pstore/apps/bioinfo/geneexpression/",
  contrastAnno = NULL
)
```

Arguments

dgeList	An DGEList object with counts, genes, and samples
designMatrix	The design matrix to model the data
contrastMatrix	The contrast matrix matching the design matrix
outdir	Output directory of the edgeR script. Default value "edgeR_output".
outfilePrefix	Prefix of the output files, for instance a reasonable name of the project, to identify the files uniquely. The files will be written in file.path(OUTDIR, 'input_data').
mps	Logical, whether molecular-phenotyping analysis is run.

limmaVoom	Logical, whether the limma-voom model is run instead of the edgeR model
appendGmt	NULL or character string, path to an additional GMT file besides the default GMT file used to perform gene-set analysis. The GMT file must exist.
debug	Logical, if TRUE, the source code of Rscript is used instead of the installed version.
rootPath	Character, the root path of the script
contrastAnno	A data.frame or NULL, contrast annotation.

Value

A character string containing the command to run the edgeR script.

Note

Following checks are done internally:

- The design matrix must have the same number of rows as the columns of the count matrix.
- The contrast matrix must have the same number of rows as the columns of the design matrix.
- Row names of the design matrix match the column names of the expression matrix. In case of suspect, the program will stop and report.

The output file names start with the outfilePrefix, followed by '-' and customed file suffixes.

Examples

```
mat <- matrix(rnbinom(100, mu=5, size=2), ncol=10)
rownames(mat) <- sprintf("gene%d", 1:nrow(mat))
myFac <- gl(2,5, labels=c("Control", "Treatment"))
y <- edgeR::DGEList(counts=mat, group=myFac)
myDesign <- model.matrix(~myFac); colnames(myDesign) <- levels(myFac)
myContrast <- limma::makeContrasts(Treatment, levels=myDesign)
edgeRcommand(y, designMatrix=myDesign, contrastMatrix=myContrast,
    outfilePrefix="test", outdir=tempdir())
## clean up
unlink(file.path(tempdir(), "input_data"), recursive=TRUE)
```

EdgeResult *Return a list of differential gene expression tables*

Description

Return a list of differential gene expression tables

Usage

```
EdgeResult(edgeObj, dgeGLM, dgeTables)
```

Arguments

edgeObj	An EdgeObject
dgeGLM	A DGEGLM object
dgeTables	A list of DGEtables.

Value

An EdgeResult object

EdgeResult-class *Object that contains test results, dgeTable, and SigFilter*

Description

Object that contains test results, dgeTable, and SigFilter

Slots

dgeGLM A DGEGLM class object that contains GLM test results

dgeTables A list of dgeTable

sigFilter Significantly regulated gene filter

EdgeSigFilter-class *Extends BaseSigFilter to filter genes base on logCPM and LR*

Description

Extends BaseSigFilter to filter genes base on logCPM and LR

Slots

logCPM Numeric, logCPM threshold (larger values are kept)

eset2DGEList *Transform an EexpressionSet to a DGEList object*

Description

Transform an EexpressionSet to a DGEList object

Usage

```
eset2DGEList(eset, groupVar = "group")
```

Arguments

eset	An ExpressionSet object
groupVar	Character string, column in phenoDatta

Value

A DGEList object

Examples

```
eset <- new("ExpressionSet",
  exprs=matrix(rpois(120, 20), nrow=20,
               dimnames=list(LETTERS[1:20], letters[1:6])),
  phenoData = new("AnnotatedDataFrame", data.frame(Sample=letters[1:6],
                 group=gl(2, 3), row.names=letters[1:6])),
  featureData = new("AnnotatedDataFrame", data.frame(Feature=LETTERS[1:20],
                 row.names=LETTERS[1:20])))
eset2DGEList(eset)
```

exampleEdgeObject *Return an example of EdgeObject*

Description

Return an example of EdgeObject

Usage

```
exampleEdgeObject(nfeat = 20, nsample = 6, ngroup = 3, lambda = 10)
```

Arguments

nfeat	Integer, number of features
nsample	Integer, number of samples
ngroup	Integer, number of groups, must be divisible by nsample.
lambda	Integer, passed to rpois to generate random counts.

Value

An EdgeObject

Examples

```
set.seed(1887) ## fix random generators
exampleEdgeObject()
exampleEdgeObject(50, 12, ngroup=4)
```

---

exportEdgeResult *Export dgeTest results*

Description

Export dgeTest results

Usage

```
exportEdgeResult(
  edgeResult,
  outRootDir,
  action = c("ask", "append", "overwrite", "no")
)
```

Arguments

edgeResult	A EdgeResult object
outRootDir	Character string, output directory
action	Character string, what happens if the output directory exists

Value

Invisibly returns NULL. Called for its side effects of writing result files to disk.

exportStaticGeneLevelPlots *Export static gene-level plots in PDF*

Description

Export static gene-level plots in PDF

Usage

```
exportStaticGeneLevelPlots(edgeResult, file)
```

Arguments

edgeResult	An EdgeResult object
file	Character string, the PDF file name

Value

Called for its side effect of writing plots to a PDF file; returns invisibly NULL.

fData,DGEList-method *Get fData*

Description

Get fData

Usage

```
## S4 method for signature 'DGEList'
fData(object)
```

Arguments

object	A DGEList

Value

A data.frame of feature (gene) annotations.

fData,EdgeObject-method
Get fData

Description

Get fData

Usage

```
## S4 method for signature 'EdgeObject'
fData(object)
```

Arguments

object	An EdgeObject

Value

A data.frame of feature (gene) annotations.

fData<-,DGEList,data.frame-method
Set fData

Description

Set fData

Usage

```
## S4 replacement method for signature 'DGEList,data.frame'
fData(object) <- value
```

Arguments

object	A DGEList
value	A data.frame

Value

The updated DGEList object with new feature annotations.

fData<-,EdgeObject,data.frame-method *Set fData*

Description

Set fData

Usage

```
## S4 replacement method for signature 'EdgeObject,data.frame'
fData(object) <- value
```

Arguments

object	An EdgeObject
value	A data.frame

Value

The updated EdgeObject with new feature annotations.

FeatAnnoExprs-class *A class that contain feature annotation and expression matrix*

Description

A class that contain feature annotation and expression matrix

Arguments

exprs	A matrix of expression
genes	A data.frame

featureNames,EdgeObject-method *Feature names*

Description

Feature names

Usage

```
## S4 method for signature 'EdgeObject'
featureNames(object)
```

Arguments

object	An EdgeObject

Value

A character vector of feature names.

filterByCPM *Filter lowly expressed genes by counts per million (CPM)*

Description

Filter lowly expressed genes by counts per million (CPM)

Usage

```
filterByCPM(obj, ...)
```

Arguments

obj	An object
...	Other parameters

Value

The return value depends on the class of obj. See method-specific documentation.

`filterByCPM.DGEList` *Filter lowly expressed genes by CPM in DGEList*

Description

Filter lowly expressed genes by CPM in DGEList

Usage

```
## S3 method for class 'DGEList'
filterByCPM(
  obj,
  minCPM = 1,
  minCount = minGroupCount(obj),
  lib.size = NULL,
  ...
)
```

Arguments

`obj`	A `DGEList` object
`minCPM`	Numeric, the minimum CPM accepted as expressed in one sample
`minCount`	Integer, how many samples must have CPM larger than `minCPM` to keep this gene?
`lib.size`	Integers of library size, or `NULL`
`...`	Not used

Value

Another `DGEList` object, with lowly expressed genes removed. The original counts and gene annotation can be found in `counts.unfiltered` and `genes.unfiltered` fields, respectively. The logical vector of the filter is saved in the `cpmFilter` field.

Examples

```
set.seed(1887)
mat <- rbind(matrix(rbinom(150, 5, 0.25), nrow=25), rep(0, 6))
d <- DGEList(mat, group=rep(1:3, each=2),
             genes=data.frame(Gene=sprintf("Gene%d", 1:nrow(mat))))
df <- filterByCPM(d)

nrow(df$counts.unfiltered) ## 26
nrow(df$counts) ## 25
```

filterByCPM.EdgeObject

Filter EdgeObj and remove lowly expressed genes

Description

Filter EdgeObj and remove lowly expressed genes

Usage

```
## S3 method for class 'EdgeObject'
filterByCPM(obj, minCPM = 1, minCount = minGroupCount(obj), ...)
```

Arguments

obj	An EdgeObject object
minCPM	Minimal CPM value, see descriptions below
minCount	Minimal count of samples in which the CPM value is no less than minCPM
...	Not used
	The filter is recommended by the authors of the edgeR package to remove lowly expressed genes, since including them in differential gene expression analysis will cause extreme differential expression fold-changes of lowly and stochastically expressed genes, and increase false positive rates.
	The filter removes genes that are less expressed than 1 copy per million reads (cpm) in at least n samples, where n equals the number of samples in the smallest group of the design.

Value

An EdgeObject with lowly expressed genes removed from the internal DGEList. The unfiltered counts and gene annotation are preserved in the counts.unfiltered and genes.unfiltered fields of the DGEList.

Examples

```
myFac <- gl(3,2)
set.seed(1234)
myMat <- matrix(rpois(1200,100), nrow=200, ncol=6)
myMat[1:3,] <- 0
myEdgeObj <- EdgeObject(myMat,
                        DesignContrast(designMatrix=model.matrix(~myFac),
                         contrastMatrix=matrix(c(0,1,0), ncol=1), groups=myFac),
                         fData=data.frame(GeneSymbol=sprintf("Gene%d", 1:200)))
myFilteredEdgeObj <- filterByCPM(myEdgeObj)
dim(counts(myEdgeObj))
dim(counts(myFilteredEdgeObj))
## show unfiltered count matrix
```

```
dim(counts(myFilteredEdgeObj, filter=FALSE))
```

`filterByCPM.matrix` *Filter lowly expressed genes by CPM*

Description

Filter lowly expressed genes by CPM

Usage

```
## S3 method for class 'matrix'
filterByCPM(obj, minCPM = 1, minCount = 1, ...)
```

Arguments

`obj`	A matrix
`minCPM`	Numeric, the minimum CPM accepted as expressed in one sample
`minCount`	Integer, how many samples must have CPM larger than `minCPM` to keep this gene?
`...`	Not used

Value

A logical vector of the same length as the row count of the matrix. `TRUE` means the gene is reasonably expressed, and `FALSE` means the gene is lowly expressed and should be filtered (removed)

Examples

```
set.seed(1887)
mat <- rbind(matrix(rbinom(125, 5, 0.25), nrow=25), rep(0, 5))
filterByCPM(mat)
```

`fitGLM` *Fit generalized linear model*

Description

Fit generalized linear model

Usage

```
fitGLM(object, ...)

## S4 method for signature 'EdgeObject'
fitGLM(object, ...)
```

Arguments

object	An EdgeObject object
...	Passed to glmFit

Value

A DGEGLM object containing the GLM fit.

The fit object

Methods (by class)

- fitGLM(EdgeObject): Method for EdgeObject

---

gctFilename *Get GCT filename from a directory*

Description

Get GCT filename from a directory

Usage

```
gctFilename(dir)
```

Arguments

dir	Character string, path to a directory where a GCT file is saved

Value

Character string, full name of the GCT file If no file is found, the function reports an error. If more than one file is found, a warning message is raised, and only the first file is used.

---

geneCount *Return gene count*

Description

Return gene count

Usage

```
geneCount(countDgeResult)
```

Arguments

countDgeResult	An EdgeResult object

Value

Integer

---

geneIdentifierTypes *Return gene identifier types*

Description

Return gene identifier types

Usage

```
geneIdentifierTypes(dgeResult)
```

Arguments

dgeResult	An DgeResult object

Value

A character string indicating the gene identifiers found The following terms are recognized: GeneID, EnsemblID, GeneSymbol, FeatureName

Note

Note that the order matters: FeatureName > GeneID = EnsemblID > GeneSymbol

---

groupCol *Get automatic group color*

Description

Get automatic group color

Usage

```
groupCol(edgeObj, panel = "Set1")
```

Arguments

edgeObj	An EdgeObject or EdgeResult object
panel	passed to fcbrewer

Value

A fcbase object

`groups,EdgeObject-method` *Get sample groups from an EdgeObject object*

Description

Get sample groups from an EdgeObject object

Usage

```
## S4 method for signature 'EdgeObject'
groups(object)
```

Arguments

`object`	An `EdgeObject` object

Value

A factor of sample group assignments.

`gtKdTable` *Print a kdTable nicely with gt*

Description

Print a kdTable nicely with gt

Usage

```
gtKdTable(kdTable, feature_label = "GeneSymbol", ...)
```

Arguments

`kdTable`	a knockdown table (kdTable) returned by `kdTable`
`feature_label`	The column which contains feature label, gene symbol by
`...`	Passed to `gt` default

Value

A `gt` table object

hasCommonDisp *Tells whether common dispersion has been set*

Description

Tells whether common dispersion has been set

Usage

```
hasCommonDisp(object)

## S4 method for signature 'DGEList'
hasCommonDisp(object)

## S4 method for signature 'EdgeObject'
hasCommonDisp(object)
```

Arguments

object	An object

Value

Logical, whether the common dispersion has been set.

Methods (by class)

- hasCommonDisp(DGEList): Method for DGEList
- hasCommonDisp(EdgeObject): Method for EdgeObject

humanGeneSymbols *Get human gene symbols for gene-set enrichment analysis*

Description

Get human gene symbols for gene-set enrichment analysis

Usage

```
humanGeneSymbols(object)

## S4 method for signature 'DGEList'
humanGeneSymbols(object)

## S4 method for signature 'EdgeObject'
humanGeneSymbols(object)
```

Arguments

object	An object

Value

A character vector of human gene symbols.

Methods (by class)

- humanGeneSymbols(DGEList): Method for DGEList
- humanGeneSymbols(EdgeObject): Method for EdgeObject

---

inferSV *Infer surrogate variables*

Description

Infer surrogate variables

Usage

```
inferSV(object, design, ...)

## S4 method for signature 'matrix,matrix'
inferSV(object, design, ...)

## S4 method for signature 'DGEList,matrix'
inferSV(object, design, ...)

## S4 method for signature 'DGEList,formula'
inferSV(object, design, ...)
```

Arguments

object	An object
design	Design matrix or formula
...	Other parameters

Value

A matrix of surrogate variables.

Methods (by class)

- `inferSV(object = matrix, design = matrix)`: method for matrix as input
- `inferSV(object = DGEList, design = matrix)`: method for voom-transformed DGEList and design matrix as input
- `inferSV(object = DGEList, design = formula)`: method for voom-transformed DGEList and formula as input

---

isAnnotated *Is the object annotated*

Description

Is the object annotated

Usage

```
isAnnotated(object)

## S4 method for signature 'EdgeObject'
isAnnotated(object)
```

Arguments

`object`	An object

Value

Logical, whether the object is annotated.

Methods (by class)

- `isAnnotated(EdgeObject)`: Method for EdgeObject

---

isEmptySV *Is the Surrogate Variable (SV) matrix empty*

Description

Is the Surrogate Variable (SV) matrix empty

Usage

```
isEmptySV(sv)
```

Arguments

sv	A surrogate variable (SV) matrix returned by sva

Value

TRUE if no valid SV was estimated; otherwise FALSE.

Examples

```
isEmptySV(matrix(0, 1,1))
isEmptySV(matrix(rnorm(5), nrow=5))
```

---

isSig *Return logical vector indicating which genes are significantly regulated*

Description

Return logical vector indicating which genes are significantly regulated

Usage

```
isSig(data.frame, sigFilter)

isSigPos(data.frame, sigFilter)

isSigNeg(data.frame, sigFilter)
```

Arguments

data.frame	A data.frame that must pass assertEdgeToptable
sigFilter	An SigFilter object

Value

A logical vector of the same length as the row number of the input data.frame

Functions

- isSigPos(): Returns which genes are significantly positively regulated
- isSigNeg(): Returns which genes are significantly negatively regulated

isUnsetAveExpr *Whether the aveExpr filter is set*

Description

Whether the aveExpr filter is set

Usage

```
isUnsetAveExpr(limmaSigFilter)
```

Arguments

limmaSigFilter	A LimmaSigFilter object

Value

Logical, whether the aveExpr threshold is the default value.

isUnsetLogCPM *Whether the logCPM filter is set*

Description

Whether the logCPM filter is set

Usage

```
isUnsetLogCPM(edgeSigFilter)
```

Arguments

edgeSigFilter	A EdgeSigFilter object

Value

Logical, whether the logCPM threshold is the default value.

isUnsetPosLogFC *Tells whether the threshold was not set*

Description

Tells whether the threshold was not set

Usage

```
isUnsetPosLogFC(sigFilter)

isUnsetNegLogFC(sigFilter)

isUnsetPValue(sigFilter)

isUnsetFDR(sigFilter)
```

Arguments

sigFilter	An SigFilter object

Value

Logical, whether the thresholds are the default values

isUnsetSigFilter *Whether the SigFilter is the default one*

Description

Whether the SigFilter is the default one

Usage

```
isUnsetSigFilter(object)
```

Arguments

object	An SigFilter object

Value

Logical, whether it is unset

kdTable *Retrieve a knockdown table from edgeRes*

Description

Retrieve a knockdown table from edgeRes

Usage

```
kdTable(edgeRes, feature, feature_label = "GeneSymbol")
```

Arguments

edgeRes	An EdgeResult object
feature	The feature to be retrieved
feature_label	The column which contains feature label, gene symbol by default

Value

A compact table containing essential information about knockdown

LimmaSigFilter-class *LimmaSigFilter Extending BaseSigFilter to filter genes base on aveExpr*

Description

LimmaSigFilter Extending BaseSigFilter to filter genes base on aveExpr

Slots

aveExpr Numeric, AveExpr threshold (larger values are kept)

LimmaVoomResult *Construct a LimmaVoomResult object*

Description

Construct a LimmaVoomResult object

Usage

```
LimmaVoomResult(edgeObj, voom, marrayLM, dgeTables)
```

Arguments

edgeObj	An EdgeObject.
voom	The voom (EList) object.
marrayLM	A MArrayLM object.
dgeTables	A list of DGEtables.

Value

An LimmaVoomResult object.

LimmaVoomResult-class *The LimmaVoom Object that contains test results, dgeTable, and SigFilter*

Description

The LimmaVoom Object that contains test results, dgeTable, and SigFilter

Slots

marrayLM A MArrayLM class object that contains results of eBayesFit

voom The voom object

logCPM *Get settings in the EdgeSigFilter*

Description

Get settings in the EdgeSigFilter

Usage

```
logCPM(edgeSigFilter)
```

Arguments

edgeSigFilter	An EdgeSigFilter object

Value

Numeric values of the logCPM filter

logFCmatrix *Extract a matrix of log2(fold-change) values*

Description

Extract a matrix of log2(fold-change) values

Usage

```
logFCmatrix(
  edgeResult,
  featureIdentifier = "GeneSymbol",
  contrasts = NULL,
  removeNAfeatures = TRUE,
  minAveExpr = NULL
)
```

Arguments

edgeResult	An EdgeResult object
featureIdentifier	Character, column name in dgeTable that will be used as rownames of the result matrix
contrasts	NULL or characters; if not NULL, only logFC values of given contrasts will be returned

removeNAfeatures
: Logical, if TRUE, features containing NA values are removed.

minAveExpr
: NULL or numeric. If set, features with aveExpr lower than the given value is not considered. This option is helpful to remove genes that are lowly expressed which yet show strong differential expression.

Value

A numeric matrix of log2(fold-change) values with features in rows and contrasts in columns.

Note

TODO: add edgeResult data example

---

logFCmatrixPCA *Perform principal component analysis to the log fold-change matrix*

Description

Perform principal component analysis to the log fold-change matrix

Usage

```
logFCmatrixPCA(lfc)
```

Arguments

lfc
: A matrix of log2 fold changes, with features in rows and contrasts in columns

Value

A PCAScoreMatrix object

The function performs principal component analysis (PCA) to the log fold-change matrix.

By using a column of zeros during the PCA analysis, which was removed from the final result, the point of origin represents an ideal contrast which yield absolutely no differential gene expression. It is easier to interpret the PCA results with this transformation.

Examples

```
my_lfc_mat <- matrix(rnorm(1000), nrow=100, ncol=10)
my_lfc_pca <- logFCmatrixPCA(my_lfc_mat)
my_lfc_pca
```

logFCpca *Perform principal component analysis to an EdgeResult object*

Description

Perform principal component analysis to an EdgeResult object

Usage

```
logFCpca(edgeResult)
```

Arguments

edgeResult	An EdgeResult object

Value

A PCAScoreMatrix object

The function performs principal component analysis (PCA) to the log fold-change matrix.

By using a column of zeros during the PCA analysis, which was removed from the final result, the point of origin represents an ideal contrast which yield absolutely no differential gene expression. It is easier to interpret the PCA results with this transformation.

See Also

logFCmatrixPCA

Examples

```
## TODO: add edgeResult sample
```

lsfEdgeR *Send an edgeR analysis job to SLF*

Description

Send an edgeR analysis job to SLF

Usage

```
lsfEdgeR(
  dgeList,
  designContrast,
  outdir = "edgeR_output",
  outfilePrefix = "an-unnamed-project-",
  overwrite = c("ask", "overwrite", "append", "no"),
  mps = FALSE,
  limmaVoom = FALSE,
  appendGmt = NULL,
  qos = c("preempt_cpu", "interactive_cpu", "batch_cpu"),
  rootPath = "/apps/rocs/pRED/groups/bioinfo/geneexpression",
  debug = FALSE
)
```

Arguments

dgeList	An DGEList object with counts, genes, and samples
designContrast	The DesignContrast object to model the data
outdir	Output directory of the edgeR script. Default value "edgeR_output".
outfilePrefix	Prefix of the output files. It can include directories, e.g. "data/outfile-". In case of NULL, temporary files will be created.
overwrite	If ask, the user is asked before an existing output directory is overwritten. If yes, the job will start and an existing directory will be overwritten anyway. If no, and if an output directory is present, the job will not be started.
mps	Logical, whether molecular-phenotyping analysis is run.
limmaVoom	Logical, whether the limma-voom model is run instead of the edgeR model.
appendGmt	NULL or character string, path to an additional GMT file for gene-set analysis. The option is passed to slurmEdgeRcommand and then to edgeRcommand.
qos	Character, specifying Quality of Service of Slurm. Available values include short (recommended default, running time cannot exceed 3 hours), interactive (useful if you wish to get the results from an interactive session), and long (useful if the job is expected to run more than three hours.) using srun and the 'interaction' queue of jobs instead of using sbatch.
rootPath	Character string, the directory of geneexpression scripts, under which bin/ngsDge_edgeR.Rscript is found.
debug	Logical, if TRUE, the source code of Rscript is used instead of the installed version. The option is passed to edgeRcommand.

Value

A list of two items, command, the command line call, and output, the output of the SLURM command in bash

Note

Even if the output directory is empty, if `overwrite` is set to `no` (or if the user answers `no`), the job will not be started.

Examples

```
mat <- matrix(rnbinom(100, mu=5, size=2), ncol=10)
rownames(mat) <- sprintf("gene%d", 1:nrow(mat))
myFac <- gl(2,5, labels=c("Control", "Treatment"))
y <- edgeR::DGEList(counts=mat, group=myFac)
myDesign <- model.matrix(~myFac); colnames(myDesign) <- levels(myFac)
myContrast <- limma::makeContrasts(Treatment, levels=myDesign)
## \dontrun{
## lsfEdgeR(y, designMatrix=myDesign, contrastMatrix=myContrast,
##  outfilePrefix="test", outdir=tempdir())
## }
```

---

lsfEdgeRcommand *Return the LSF command to run the edgeR script*

Description

Return the LSF command to run the edgeR script

Usage

```
lsfEdgeRcommand(
  dgeList,
  designContrast,
  outdir = "edgeR_output",
  outfilePrefix = "an-unnamed-project-",
  mps = FALSE,
  limmaVoom = FALSE,
  appendGmt = NULL,
  qos = c("long", "preempty", "short"),
  rootPath = "/apps/rocs/pRED/groups/bioinfo/geneexpression",
  debug = FALSE,
  bsubFile = NULL
)
```

Arguments

`dgeList`	An `DGEList` object with `counts`, `genes`, and `samples`
`designContrast`	The DesignContrast object to model the data
`outdir`	Output directory of the edgeR script. Default value "edgeR_output".

outfilePrefix	Prefix of the output files. It can include directories, e.g. "data/outfile-". In case of NULL, temporary files will be created.
mps	Logical, whether molecular-phenotyping analysis is run.
limmaVoom	Logical, whether the limma-voom model is run instead of the edgeR model
appendGmt	NULL or character string, path to an additional GMT file for gene-set analysis. The option is passed to edgeRcommand.
qos	Character, specifying Quality of Service of LSF Available values include long, short, and preempty.
rootPath	Character string, the directory of geneexpression scripts, under which bin/ngsDge_edgeR.Rscript is found.
debug	Logical, if TRUE, the source code of Rscript is used instead of the installed version. The option is passed to edgeRcommand.
bsubFile	NULL or character string, file name that contains LSF jobs. If NULL, a file will be generated within the current directory following the pattern of outfilePrefix.bsub.
	This function wraps the function edgeRcommand to return the command needed to start a LSF job.
	It uses outdir to specify slurm output and error files as in the same directory of outdir. And the job name is set as the name of the output directory.

Value

A character string containing the LSF bsub command to submit the edgeR analysis job.

See Also

edgeRcommand

Examples

```
mat <- matrix(rnbinom(100, mu=5, size=2), ncol=10)
rownames(mat) <- sprintf("gene%d", 1:nrow(mat))
myFac <- gl(2,5, labels=c("Control", "Treatment"))
y <- edgeR::DGEList(counts=mat, group=myFac)
myDesign <- model.matrix(~myFac); colnames(myDesign) <- levels(myFac)
myContrast <- limma::makeContrasts(Treatment, levels=myDesign)
myDesCon <- DesignContrast(designMatrix=myDesign, contrastMatrix=myContrast)
lsfEdgeRcommand(y, designContrast=myDesCon,
    outfilePrefix="test", outdir=tempdir())
## clean up
unlink(file.path(tempdir(), "input_data"), recursive=TRUE)
unlink("test.bsub")
```

mergeDGEList *Merge two DGEList objects into one*

Description

Merge two DGEList objects into one

Usage

```
mergeDGEList(firstDgeList, secondDgeList, DGEListLabels = NULL)
```

Arguments

firstDgeList	First DGEList object
secondDgeList	Second DGEIst object
DGEListLabels	Labels, either NULL or a vector of character strings with length two

The function merges two DGEList objects. It does essentially three things:

Feature annotation It extracts the common features from both objects, and use the feature annotation in the firstDgeList object as the annotation for the final object.

Sample annotation It extracts the common columns from sample annotation of both objects, and row-bind them as the annotation for the final object.

counts Matching final features and samples, the counts matrices are column-binded.

In case DGEListLabels is available, its values will be turned into a factor vector, and appended as the column DGEListLabel in the samples object of the returned value.

Value

A DGEList object containing the merged counts, sample annotation, and gene annotation from both input objects.

Examples

```
y1 <- matrix(rnbinom(1000, mu=5, size=2), ncol=4)
genes1 <- data.frame(GeneSymbol=sprintf("Gene%d", 1:nrow(y1)))
rownames(y1) <- rownames(genes1) <- 1:nrow(y1)
anno1 <- data.frame(treatment=gl(2,2, labels=c("ctrl", "tmt")),
    donor=factor(rep(c(1,2), each=2)))
d1 <- DGEList(counts=y1, genes=genes1, samples=anno1)

y2 <- matrix(rnbinom(1000, mu=5, size=2), ncol=4)
genes2 <- data.frame(GeneSymbol=sprintf("Gene%d", 1:nrow(y2)+100))
rownames(y2) <- rownames(genes1) <- 1:nrow(y2)+100
anno2 <- data.frame(treatment=gl(2,2, labels=c("ctrl", "tmt")),
    sex=factor(rep(c("m", "f"), each=2)))
```

```
d2 <- DGEList(counts=y2, genes=genes2, samples=anno2)

md <- mergeDGEList(d1, d2)
md2 <- mergeDGEList(d1, d2, DGEListLabels=c("d1", "d2"))
```

`minGroupCount` *Return the size of the smallest group*

Description

Return the size of the smallest group

Usage

```
minGroupCount(obj)

## S3 method for class 'DGEList'
minGroupCount(obj)

## S3 method for class 'EdgeObject'
minGroupCount(obj)
```

Arguments

`obj`	A `DGEList` or `EdgeObject` object

Value

Integer

Methods (by class)

- `minGroupCount(DGEList)`: Return the size of the smallest group defined in the `DGEList` object
- `minGroupCount(EdgeObject)`: Return the size of the smallest group defined in the `EdgeObject` object

Examples

```
y <- matrix(rnbinom(12000,mu=10,size=2),ncol=6)
d <- DGEList(counts=y, group=rep(1:3,each=2))
minGroupCount(d) ## 2
d2 <- DGEList(counts=y, group=rep(1:2,each=3))
minGroupCount(d2) ## 3
d3 <- DGEList(counts=y, group=rep(1:3, 1:3))
minGroupCount(d3) ## 1
```

model.DGEList *Build design matrix from a DGEList object*

Description

Build design matrix from a DGEList object

Usage

```
model.DGEList(object, formula, ...)
```

Arguments

object	A DGEList object
formula	Formula, passed to model.matrix Sample annotation is used to construct the formula
...	Not used so far

Value

A design matrix.

modLogCPM *Modulated logCPM*

Description

Modulated logCPM

Usage

```
modLogCPM(object, ...)

## S4 method for signature 'DGEList'
modLogCPM(object, prior.count = 2)

## S4 method for signature 'EdgeObject'
modLogCPM(object)
```

Arguments

object	A DGEList object
...	Other parameters
prior.count	Integer, prior count.

Value

A numeric matrix of modulated log-CPM values.

Methods (by class)

- `modLogCPM(DGEList)`: Method for DGEList
- `modLogCPM(EdgeObject)`: Method for EdgeObject

`naOrSqrt` *Return either NA (if input is NULL) or sqrt*

Description

Return either NA (if input is NULL) or sqrt

Usage

```
naOrSqrt(x)
```

Arguments

`x`	Numeric value

Value

`NA` or sqrt value

`ncol,EdgeResult-method` *Return number of samples*

Description

Return number of samples

Usage

```
## S4 method for signature 'EdgeResult'
ncol(x)
```

Arguments

`x`	An EdgeResults object

Value

Integer

`nContrast,EdgeResult-method`
Return the number of contrasts

Description

Return the number of contrasts

Usage

```
## S4 method for signature 'EdgeResult'
nContrast(object)
```

Arguments

`object`	An `EdgeResult` object.

Value

An integer, the number of contrasts.

`normalize,EdgeObject-method`
Normalize an EdgeObject

Description

Normalize an EdgeObject

Usage

```
## S4 method for signature 'EdgeObject'
normalize(object, method = "RLE", ...)
```

Arguments

`object`	An `EdgeObject` object.
`method`	Method passed to `calcNormFactors`.
`...`	Other parameters passed to `calcNormFactors`.

Value

An `EdgeObject` with updated normalization factors in the internal `DGEList`.

Methods (by class)

- `normalize(EdgeObject)`: Normalize an EdgeObject by calculating normalization factors using the given method

---

normBoxplot *Plot distribution of normalized counts*

Description

Plot distribution of normalized counts

Usage

```
normBoxplot(before.norm, after.norm, ...)
```

Arguments

`before.norm`	An `EdgeObject` before normalization.
`after.norm`	An `EdgeObject` after normalization.
`...`	Other parameters passed to `boxplot`.

Value

Called for its side effect of plotting; returns invisibly NULL.

---

normFactors *Extract normalisation factors from the object*

Description

Extract normalisation factors from the object

Usage

```
normFactors(object)

## S4 method for signature 'DGEList'
normFactors(object)

## S4 method for signature 'EdgeObject'
normFactors(object)
```

Arguments

`object`	An object

Value

A numeric vector of normalisation factors.

Methods (by class)

- `normFactors(DGEList)`: Method for DGEList
- `normFactors(EdgeObject)`: Method for EdgeObject

`nrow,EdgeResult-method` *Return number of features*

Description

Return number of features

Usage

```
## S4 method for signature 'EdgeResult'
nrow(x)
```

Arguments

`x`	An EdgeResults object

Value

Integer

`pairs.EdgeResult` *Pairs plot for EdgeResult*

Description

Pairs plot for EdgeResult

Usage

```
## S3 method for class 'EdgeResult'
pairs(
  x,
  lower.panel = panel.lmSmooth,
  upper.panel = panel.cor,
  freeRelation = TRUE,
  pch = 19,
  ...
)
```

Arguments

x	An EdgeResult object.
lower.panel	Lower panel, passed to pairs.
upper.panel	Upper panel, passed to pairs.
freeRelation	Logical, whether x- and y-axis shoule have the same range
pch	Point symbol
...	passed to pairs
	Plot pairwise logFCs

Value

Called for its side effect of plotting; returns invisibly NULL.

See Also

pairs.

---

parseMolPhenFeat *Parse feature information from molecular-phenotyping GCT files*

Description

Parse feature information from molecular-phenotyping GCT files

Usage

```
parseMolPhenFeat(gctMatrix)
```

Arguments

gctMatrix	A GctMatrix capturing the counts of a molecular phenotyping experiment

Value

A data.frame with following columns: GeneID (as integer), GeneSymbol (as character), and Transcript, with the original names as row names

See Also

readMolPhenCoverageGct, which calls this function internally to parse molecular phenotyping gene features

pData,DGEList-method *Get pData (sample annotation)*

Description

Get pData (sample annotation)

Usage

```
## S4 method for signature 'DGEList'
pData(object)
```

Arguments

object	A DGEList

Value

A data.frame of sample annotations.

pData,EdgeObject-method *Get pData*

Description

Get pData

Usage

```
## S4 method for signature 'EdgeObject'
pData(object)
```

Arguments

object	An EdgeObject

Value

A data.frame of sample annotations.

`pData<-,DGEList,data.frame-method`
Set pData (sample annotation)

Description

Set pData (sample annotation)

Usage

```
## S4 replacement method for signature 'DGEList,data.frame'
pData(object) <- value
```

Arguments

`object`	A DGEList
`value`	A `data.frame`

Value

The updated `DGEList` object with new sample annotations.

`pData<-,EdgeObject,data.frame-method`
Set pData (sample annotation)

Description

Set pData (sample annotation)

Usage

```
## S4 replacement method for signature 'EdgeObject,data.frame'
pData(object) <- value
```

Arguments

`object`	A DGEList
`value`	A `data.frame`

Value

The updated `EdgeObject` with new sample annotations.

plotBCV *Plot BCV*

Description

Plot BCV

Usage

```
plotBCV(x, ...)

## S4 method for signature 'DGEList'
plotBCV(x, ...)

## S4 method for signature 'EdgeObject'
plotBCV(x, ...)

## S4 method for signature 'EdgeResult'
plotBCV(x, ...)
```

Arguments

x	An object
...	Other paramters

Value

Called for its side effect of plotting; returns invisibly NULL.

Methods (by class)

- `plotBCV(DGEList)`: Method for DGEList
- `plotBCV(EdgeObject)`: Method for EdgeObject
- `plotBCV(EdgeResult)`: Method for EdgeResult

plotKnockdown *Plot gene expression with knockdown efficiency*

Description

Plot gene expression with knockdown efficiency

Usage

```
plotKnockdown(
  goiExpr,
  exprsVar = "exprs",
  groupVar = "group",
  controlGroup = NULL,
  trans = "identity",
  exprsUnit = "Arbitrary Unit",
  test = c("wilcox.test", "t.test")
)
```

Arguments

goiExpr	A data.frame containing expression of gene of interest in linear scale, which must contain columns given below as exprsVar and groupVar.
exprsVar	Character, the variable name of expression. The unit must be in linear scale, not in logarithmic scale, otherwise the knockdown efficiency calculation will be wrong.
groupVar	Character, the variable name of grouping. The column must be a factor or a character.
controlGroup	NULL or character. If groupVar is a factor and controlGroup is NULL, then the first level is assumed to be the control. Otherwise, controlGroup must be in the group variable.
trans	Character, transformation of the y-axis, commonly used values include identity (do not transform), log10, and log2.
exprsUnit	Character, unit name of expression (TPM, CPM, RPKM, etc.)
test	Character, statistical test, wilcox.test and t.test are supported.

Value

A ggplot object displaying boxplots of gene expression across groups with a secondary axis showing knockdown efficiency.

Examples

```
myData <- data.frame(group=gl(3,4),
 exprs=as.vector(sapply(c(100, 10, 1), function(x) rnorm(4, x))))
plotKnockdown(myData)

myData2 <-  data.frame(group=rep(c("Vehicle", "Dose1", "Dose2"), each=4),
 exprs=as.vector(sapply(c(100, 10, 1), function(x) rnorm(4, x))))
plotKnockdown(myData2, controlGroup="Vehicle")
```

plotMDS.EdgeObject *plotMDS for EdgeObject*

Description

plotMDS for EdgeObject

Usage

```
## S3 method for class 'EdgeObject'
plotMDS(x, ...)
```

Arguments

x	An EdgeObject object
...	Other parameters passed to plotMDS.

Value

An MDS object (invisibly), as returned by plotMDS.

plotTopSigGenes *Plot top significantly differentially expressed genes by contrast*

Description

Plot top significantly differentially expressed genes by contrast

Usage

```
plotTopSigGenes(
  countDgeResult,
  n = 5,
  nSigned = NULL,
  identifier = "GeneSymbol"
)
```

Arguments

countDgeResult	A CountDgeResult object, for instance EdgeResult or LimmaVoomDgeResult.
n	Integer, how many genes should be visualized per contrast.
nSigned	NULL or integer, in the later case the top nSigned genes from positively and negatively regulated genes are shown per contrast, respectively.
identifier	Character string, column name in genes annotation to be used to index and display the genes.

Value

A ggplot object

See Also

plotTopSigGenesByContrast plots one contrast at a time.

Examples

```
edgeObj <- exampleEdgeObject()
edgeRes <- dgeWithEdgeR(edgeObj)
plotTopSigGenes(edgeRes, n=6)
## display top two positive and top three negative genes
plotTopSigGenes(edgeRes, nSigned=2)
```

---

plotTopSigGenesByContrast

Plot top significantly differentially expressed genes by contrast

---

Description

Plot top significantly differentially expressed genes by contrast

Usage

```
plotTopSigGenesByContrast(
  countDgeResult,
  contrast,
  n = 5,
  nSigned = NULL,
  identifier = "GeneSymbol"
)
```

Arguments

countDgeResult	A CountDgeResult object, for instance EdgeResult or LimmaVoomDgeResult.
contrast	A character string, or an index (integer), or a logical vector with one TRUE element, to indicate which contrast to plot
n	Integer, how many genes should be visualized.
nSigned	NULL or integer, in the later case the top nSigned genes from positively and negatively regulated genes are shown, respectively.
identifier	Character string, column name in genes annotation to be used to index and display the genes.

Value

A ggplot object. If nSigned is not NULL, genes are plotted with colors: blue indicating down-regulated genes and red indicating up-regulated genes.

See Also

plotTopSigGenesByContrast plots all contrasts at once.

Examples

```
edgeObj <- exampleEdgeObject()
edgeRes <- dgeWithEdgeR(edgeObj)
plotTopSigGenesByContrast(edgeRes, n=6, contrast=1)
plotTopSigGenesByContrast(edgeRes, n=6, contrast=2)
## display top three positive and top three negative genes
plotTopSigGenesByContrast(edgeRes, nSigned=3, contrast=1)
plotTopSigGenesByContrast(edgeRes, nSigned=4, contrast=2)
```

---

posLogFC *Get settings in the significance filter*

Description

Get settings in the significance filter

Usage

```
posLogFC(sigFilter)

negLogFC(sigFilter)

pValue(sigFilter)

FDR(sigFilter)
```

Arguments

sigFilter	An SigFilter object

Value

Numeric values of the thresholds

posLogFC<- *Update SigFilter*

Description

Update SigFilter

Usage

```
posLogFC(object) <- value

negLogFC(object) <- value

logFC(object) <- value

pValue(object) <- value

FDR(object) <- value

logCPM(object) <- value

aveExpr(object) <- value

## S3 method for class 'SigFilter'
update(object, logFC, posLogFC, negLogFC, pValue, FDR, ...)

## S3 method for class 'EdgeSigFilter'
update(object, logFC, posLogFC, negLogFC, pValue, FDR, logCPM, ...)

## S3 method for class 'LimmaSigFilter'
update(object, logFC, posLogFC, negLogFC, pValue, FDR, aveExpr, ...)
```

Arguments

object	An SigFilter object
value	Numeric, vssigned threshold value
logFC	Numeric, logFC filter value, optional.
posLogFC	Numeric, positive logFC filter value, optional.
negLogFC	Numeric, negative logFC filter value, optional.
pValue	Numeric, pValue filter value, optional
FDR	Numeric, FDR filter value, optional
...	not used now
logCPM	Numeric, logCPM filter value, optional (only for EdgeSigFilter).
aveExpr	Numeric, aveExpr filter value, optional (only for LimmaSigFilter).

Value

An updated `SigFilter` object.

An updated `EdgeSigFilter` object.

An updated `LimmaSigFilter` object.

Functions

- `posLogFC(object) <- value`: Updates the posLogFC threshold value
- `negLogFC(object) <- value`: Updates the negLogFC threshold value
- `logFC(object) <- value`: Updates the posLogFC threshold value
- `pValue(object) <- value`: Updates the pValue threshold value
- `FDR(object) <- value`: Updates the FDR threshold value
- `logCPM(object) <- value`: Updates the logCPM threshold value
- `aveExpr(object) <- value`: Updates the aveExpr threshold value

`prcomp.DGEList` *Principal component analysis of DGEList*

Description

Principal component analysis of DGEList

Usage

```
## S3 method for class 'DGEList'
prcomp(x, ntop = NULL, scale = FALSE, verbose = FALSE, ...)
```

Arguments

`x`	A `DGEList` object
`ntop`	Integer, how many top-variable features should be used? If `NULL`, all features are used
`scale`	Logical, whether variance of features should be scaled to 1. `FALSE` by default (recommended!); set it to `TRUE` only if you are sure what you are doing
`verbose`	Logical, whether the function should print messages.
`...`	Other parameters passed to `vsnMatrix` The function first remove all-zero-count features, because they can make the PCA plot of samples delusive. Next, it applies `vsn` transformation implemented in the `vsn` package to the count matrix. Finally, PCA is applied to the vsn-transformed matrix.

Value

The function returns a prcomp object. The fit object is saved in the vsnFit field in the returned object, and the transformed matrix is saved in the vsnMat field.

See Also

prcompExprs

Examples

```
myCounts <- matrix(rnbinom(10000, 3, 0.25), nrow=1000)
myDgeList <- DGEList(counts=myCounts,
  samples=data.frame(group=gl(5,2)))
myPrcomp <- prcomp(myDgeList)


  vsn::meanSdPlot(myPrcomp$vsnFit)


## features with zero count in all samples do not contribute to the PCA analysis
myDgeList2 <- DGEList(counts=rbind(myCounts, rep(0, 10)),
  samples=data.frame(group=gl(5,2)))
myPrcomp2 <- prcomp(myDgeList2)
stopifnot(identical(myPrcomp, myPrcomp2))
```

---

prcomp.DGEListList *Run principal component analysis on a DGEListList object*

Description

Run principal component analysis on a DGEListList object

Usage

```
## S3 method for class 'DGEListList'
prcomp(x, ntop = NULL, fun = function(x) cpm(x, log = TRUE), ...)
```

Arguments

x	A DGEListList object
ntop	NULL or integer. If set, only ntop top-variable genes are used
fun	Function, used to transform count data into continuous data used by PCA
...	Not used.

Value

A list of prcomp objects.

prcompExprs *Principal component analysis of an expression matrix*

Description

Principal component analysis of an expression matrix

Usage

```
prcompExprs(matrix, ntop = NULL, scale = FALSE, nbin = NULL)
```

Arguments

matrix	Numeric matrix. Features in rows and samples in columns.
ntop	Integer or NULL. If not NULL, only ntop genes with the highest variance are used for the calculation.
scale	Logical, whether variance of features should be scaled to 1. Default FALSE, as recommended by Nguyen et al. (2019)
nbin	Integer. Genes are divided into nbin bins by their average gene expression signal, and top variable genes (approximately ntop/nbin) are selected from each bin. If NULL or NA, an automatic value (100, or nrow(matrix) %/% 10 when fewer are 1000 genes are used as input) is used. It is only used when ntop is not NULL.

Value

A prcomp object.

References

Nguyen, Lan Huong, and Susan Holmes. "Ten Quick Tips for Effective Dimensionality Reduction." PLOS Computational Biology 15, no. 6 (2019): e1006907

See Also

topVarRowsByMeanBinning

Examples

```
myTestExprs <- matrix(rnorm(1000), ncol=10, byrow=FALSE)
myTestExprs[1:50, 6:10] <- myTestExprs[1:50, 6:10] + 2
myTopPca <- prcompExprs(myTestExprs, ntop=50, nbin=5)
```

pseudoTfromPvalue *Convert p-values to t-statistics*

Description

Convert p-values to t-statistics

Usage

```
pseudoTfromPvalue(p, df, sign, replaceZero = TRUE)
```

Arguments

p	Numeric, a numeric vector between 0 and 1.
df	Numeric, degree of freedom.
sign	Logical or integer, positive numbers or TRUE are interpreted as positive, and negative numbers or TRUE are interpreted as negative.
replaceZero	Logical, whether small p values or 0 should be replaced by a sufficient small number. Default and recommended: TRUE

Value

A numeric vector of pseudo t-statistics.

Examples

```
pVals <- 10^(seq(-11,0))
signs <- rep(c(TRUE, FALSE), 6)
tVals <- pseudoTfromPvalue(pVals, 5, sign=signs)
logFCs <- rep(c(1.2,-1.2),6)
tValsLogFCs <- pseudoTfromPvalue(pVals, 5, sign=logFCs)
```

quantileRange *Return a range determined by the quantile of the data*

Description

Return a range determined by the quantile of the data

Usage

```
quantileRange(x, outlier = 0.01, symmetric = TRUE)
```

Arguments

x	A numeric vector
outlier	Quantile (lower and higher) threshold
symmetric	Logical, whether the range must be symmetric around zero.

Value

A numeric vector of two (c(low, high)).

---

read_illumina_sampleSheet_xls *Read Illumina MolPhen sample sheet from XLS files*

Description

Read Illumina MolPhen sample sheet from XLS files

Usage

```
read_illumina_sampleSheet_xls(file)
```

Arguments

file	A XLS/XLSX file containing in the first sheet the sample sheet of a molecular phenotyping experiment

Value

A data.frame annotating the samples

---

readBiokitAsDGEList *Read a Biokit output directory into a DGEList object for downstream analysis*

Description

Read a Biokit output directory into a DGEList object for downstream analysis

Usage

```
readBiokitAsDGEList(
  dir,
  anno = c("refseq", "ensembl", "gencode"),
  useCollapsedData = FALSE,
  verbose = FALSE
)
```

Arguments

dir	Biokit output directory
anno	Annotation type, either refseq or ensembl is supported
useCollapsedData	Logical, FALSE as default. If set to TRUE, counts are collapsed by gene symbols. This is not recommended because gene symbols are not stable identifiers.
verbose	Logical The function depends on gct (gct-ens) and annot directories of biokit output directory.

Value

A DGEList object with count and TPM matrices, sample annotation, feature annotation, and an additional BiokitAnno element indicating the annotation type used.

Examples

```
##... (TODO: add a mock output directory in testdata)
```

---

readBiokitFeatureAnnotation

Read feature annotation from Biokit directory

Description

Read feature annotation from Biokit directory

Usage

```
readBiokitFeatureAnnotation(
  dir,
  anno = c("refseq", "ensembl", "gencode"),
  verbose = FALSE
)
```

Arguments

dir	Character string, a Biokit output directory.
anno	Character, indicating the annotation type.
verbose	Logical

Value

A `data.frame` containing feature annotation, with feature IDs as characters in rownames. The data frame contains following columns depending on the `anno` parameter:

1. FeatureName, the primary key of feature name as characters
2. GeneID (refseq only) or EnsemblID (ensembl only)
3. GeneSymbol
4. mean: mean length
5. median: median length
6. longest_isoform: longest isoform
7. merged: total length of merged exons

The function depends on the `refseq.annot.gz` (`ensembl.annot.gz`) and `refseq.geneLength.gz` (`ensembl.geneLength.gz`) files in the biokit directory.

If `.annot.gz` file is not found (which can be the case, for instance, when older biokit output directories are used), feature annotation is read from the count GCT file. The resulting `data.frame` will only contain two columns: `FeatureName` and `Description`.

If `.geneLength.gz` file is not found, no gene length information is appended.

Examples

```
## TODO add small example files
```

`readBiokitGctFile` *Read GCT files from Biokit output directory*

Description

Read GCT files from Biokit output directory

Usage

```
readBiokitGctFile(
  dir,
  anno = c("refseq", "ensembl", "gencode"),
  type = c("count", "tpm", "count_collapsed", "tpm_collapsed", "log2tpm"),
  verbose = FALSE
)
```

Arguments

`dir`	Biokit output directory
`anno`	Annotation type, either `refseq`, `ensembl`, and `gencode` is supported
`type`	GCT file type, `count`, `tpm`, `count_collapsed`, `tpm_collapsed`, and `log2tpm` are supported.
`verbose`	Logical, if TRUE, verbose mode is turned on.

Value

A numeric matrix with the attribute desc encoding the values in the description column of the GCT format.

The function depends on gct (in case anno="refseq") or gct-ens (in case anno="ensembl") subdirectory in the biokit output directory.

Examples

```
##... (TODO: add a mock output directory in testdata)
```

---

readBiokitPhenodata *Read Biokit phenodata*

Description

Read Biokit phenodata

Usage

```
readBiokitPhenodata(dir, verbose = FALSE)
```

Arguments

dir	Character string, Biokit output directory
verbose	Logical

Value

A data.frame with sample annotation in columns, and sample names (identical as the names in gct files, character strings) are row names. Nmes of the first three columns are fixed:

1. SampleName, SampleID and group concatenated by underscore
2. SampleID
3. group

The function depends on the annot/phenoData.meta file in the biokit output directory.

readFeatureAnnotationForEdgeR *Read feature annotation for EdgeR pipeline*

Description

Read feature annotation for EdgeR pipeline

Usage

```
readFeatureAnnotationForEdgeR(featureNames, file = NULL)
```

Arguments

featureNames	Character string, feature names
file	A tab-delimited file with header that provides feature annotation

Value

A data.frame, with the first column named FeatureName, which are the input feature names. The rest columns contain annotations.

The functions tries to parse feature annotation file if it is present. If not, it will use guessAndAnnotate to annotate the features.

Note that for gene symbols, only human gene symbols are supported.

Examples

```
anno <- "GeneID\tGeneSymbol\n1234\tCCR5\n1235\tCCR6"
annoFile <- tempfile()
writeLines(anno, annoFile)

featIds <- c("1235", "1234")
## use file
readFeatureAnnotationForEdgeR(featIds, file=annoFile)
## Not run:
  ## use ribiosAnnotation, depending on database connection
  readFeatureAnnotationForEdgeR(featIds, file=NULL)

## End(Not run)
```

readMolPhenAsDGEList *Read molecular phenotyping output folder into a DGEList object*

Description

Read molecular phenotyping output folder into a DGEList object

Usage

```
readMolPhenAsDGEList(dir)
```

Arguments

dir	Path of molecular phenotyping output folder, generated by the mpsnake tool.

Value

A DGEList object containing counts, gene and sample annotation.

Examples

```
#todo
```

readMolPhenCoverageGct *Read molecular phenotyping coverage file*

Description

Read molecular phenotyping coverage file

Usage

```
readMolPhenCoverageGct(file)
```

Arguments

file	Character string, a coverage GCT file of a molecular phenotyping experiment.

Value

A list of two elements: coverage, which represents the coverage matrix, and genes, which represents feature annotation.

Examples

```
mpsCov <- readMolPhenCoverageGct(system.file(file.path("extdata",
    "AmpliSeq_files",
    "MolPhen-coverage-example-20200115.gct"),
  package="ribiosNGS"))
```

`readMpsnakeAsDGEList` *Read mpsnake output directory into a DGEList object*

Description

Read mpsnake output directory into a DGEList object

Usage

```
readMpsnakeAsDGEList(dir, minReads = NULL)
```

Arguments

`dir`	Character string, path of mpsnake pipeline directory (or the `results` subdirectory).
`minReads`	`NULL` or tnteger, minimalistic read numbers for a sample to be considered. In case of `NULL`, no filtering is performed.

Value

A `DGEList` object containing counts, gene, and sample annotation

Examples

```
mpsnakeDir <- system.file("extdata/mpsnake-minimal-outdir", package="ribiosNGS")
mpsDgeList <- readMpsnakeAsDGEList(mpsnakeDir)

## equivalent
mpsnakeResDir <- system.file("extdata/mpsnake-minimal-outdir", "results",
  package="ribiosNGS")
mpsDgeList <- readMpsnakeAsDGEList(mpsnakeResDir)
```

readSampleAnnotationForEdgeR *Read sample annotation from tab-delimited file for EdgeR analysis*

Description

Read sample annotation from tab-delimited file for EdgeR analysis

Usage

```
readSampleAnnotationForEdgeR(sampleNames, file = NULL, ...)
```

Arguments

sampleNames	Character string, giving sample names
file	Character string, path to a tab-delimited file, or NULL. The first column must be either row names (namely no colum name), or sample names in the same order of sampleNames.
...	Other parameter passed to read.table.

Value

A data.frame containing sample annotation, removing 'lib.size', and 'norm.factors' because they will be added by the edgeR pipeline

Examples

```
phenoDataFile <- system.file("extdata/phenoData/test-phenoData.txt",
  package="ribiosNGS")
readSampleAnnotationForEdgeR(phenoDataFile)
readSampleAnnotationForEdgeR(file=NULL, sampleNames=as.character(1:4))
```

renameContrast *Rename contrast by a pair of vectors*

Description

Rename contrast by a pair of vectors

Usage

```
renameContrast(edgeResult, oldContrastName, newContrastName)
```

Arguments

edgeResult	An EdgeResult object
oldContrastName	A vector of character strings giving old contrast names
newContrastName	completeA vector of character strings giving new contrast names, which match the oldContrastName one to one.

Value

A new EdgeResult object

---

renameContrastByFunc *Rename contrast by a function*

Description

Rename contrast by a function

Usage

```
renameContrastByFunc(edgeResult, func)
```

Arguments

edgeResult	An EdgeResult object
func	A function receiving a vector of character strings as input, and returns another vector of the same length as output, for instance gsub. The function can be called to rename contrasts

Value

A new EdgeResult object

replaceNAwithZero *Replace NA counts with zero counts*

Description

Replace NA counts with zero counts

Usage

```
replaceNAwithZero(edgeObj)
```

Arguments

edgeObj	An EdgeObject object

Value

An EdgeObject object

ribiosNGS *_PACKAGE*

Description

ribiosNGS provides data structures and functions for next-generation sequencing gene expression analysis

rowVars *Variance of features in rows*

Description

Variance of features in rows

Usage

```
rowVars(x, na.rm = TRUE)
```

Arguments

x	Numeric matrix
na.rm	Logical. Should missing values (including NaN) be omitted from the calculations?

Value

A numeric vector of row variances.

Examples

```
myVal <- matrix(1:9, nrow=3, byrow=FALSE)
myVar <- rowVars(myVal)
stopifnot(identical(myVar, c(9,9,9)))
```

`rpkm2tpm` *Convert a RPKM matrix to a TPM matrix*

Description

Convert a RPKM matrix to a TPM matrix

Usage

```
rpkm2tpm(x)
```

Arguments

`x`	A count matrix or other objects that can be converted to a matrix by `as.matrix`

Value

transcripts per million (TPM) values

See Also

`rpkm2tpm`

Examples

```
testMatrix <- matrix(rnbinom(200, size=5, prob=0.1), nrow=20, ncol=10)
testMatrixGeneLen <- as.integer(10^rnorm(20, mean=3, sd=0.5))
testMatrixTpm <- tpm(testMatrix, testMatrixGeneLen)
testMatrixRpkm <- edgeR::rpkm(testMatrix, testMatrixGeneLen)
testthat::expect_equal(testMatrixTpm, rpkm2tpm(testMatrixRpkm))
```

sampleNames,DGEList-method

Return sample names from a DGEList object

Description

Return sample names from a DGEList object

Usage

```
## S4 method for signature 'DGEList'
sampleNames(object)
```

Arguments

object	A DGEList object

Value

A character vector of sample names.

sampleNames,EdgeObject-method

Sample names

Description

Sample names

Usage

```
## S4 method for signature 'EdgeObject'
sampleNames(object)
```

Arguments

object	An EdgeObject

Value

A character vector of sample names.

setCommonDispIfMissing *Set common dispersion if missing*

Description

Set common dispersion if missing

Usage

```
setCommonDispIfMissing(object, value)

## S4 method for signature 'DGEList,numeric'
setCommonDispIfMissing(object, value)

## S4 method for signature 'EdgeObject,numeric'
setCommonDispIfMissing(object, value)
```

Arguments

object	An object
value	Numeric

Value

The object, with common dispersion set if it was previously missing.

Methods (by class)

- setCommonDispIfMissing(object = DGEList, value = numeric): Method for DGEList
- setCommonDispIfMissing(object = EdgeObject, value = numeric): Method for EdgeObject

show,DGEList-method *Show DGEList*

Description

Show DGEList

Usage

```
## S4 method for signature 'DGEList'
show(object)
```

Arguments

object	A DGEList object

Value

Called for its side effect of printing; returns invisibly NULL.

---

show,DGEListList-method

Show DGEListList

Description

Show DGEListList

Usage

```
## S4 method for signature 'DGEListList'
show(object)
```

Arguments

object	A DGEListList object

Value

Called for its side effect of printing; returns invisibly NULL.

---

show,EdgeResult-method

Show an EdgeResult object

Description

Show an EdgeResult object

Usage

```
## S4 method for signature 'EdgeResult'
show(object)
```

Arguments

object	An EdgeResult object

Value

Invisibly returns the formatted message string.

---

show,EdgeSigFilter-method
Show an EdgeSigFilter object

Description

Show an EdgeSigFilter object

Usage

```
## S4 method for signature 'EdgeSigFilter'
show(object)
```

Arguments

object	An SigFilter object

Value

Invisibly returns the formatted message strings.

---

show,LimmaSigFilter-method
Show an LimmaSigFilter object

Description

Show an LimmaSigFilter object

Usage

```
## S4 method for signature 'LimmaSigFilter'
show(object)
```

Arguments

object	An LimmaSigFilter object

Value

Invisibly returns the formatted message strings.

show,SigFilter-method *Show an SigFilter object*

Description

Show an SigFilter object

Usage

```
## S4 method for signature 'SigFilter'
show(object)
```

Arguments

object	An SigFilter object

Value

Invisibly returns the formatted message string.

sigFilter *Retrieve SigFilter objects from other objects Return the SigFilter in use*

Description

Retrieve SigFilter objects from other objects Return the SigFilter in use

Usage

```
sigFilter(countDgeResult)
```

Arguments

countDgeResult	An countDgeResult object

Value

An SigFilter object

SigFilter *Build a SigFilter*

Description

Build a SigFilter

Usage

```
SigFilter(logFC, posLogFC, negLogFC, pValue, FDR)

EdgeSigFilter(logFC, posLogFC, negLogFC, pValue, FDR, logCPM)

LimmaSigFilter(logFC, posLogFC, negLogFC, pValue, FDR, aveExpr)
```

Arguments

logFC	Missing or positive numeric
posLogFC	Missing or positive numeric
negLogFC	Missing or negative numeric
pValue	Missing or numeric between 0 and 1
FDR	Missing or numeric between 0 and 1
logCPM	logCPM filter, only valid for EdgeSigFilter
aveExpr	Average expression filter, only valid for LimmaSigFilter

Value

A SigFilter object

Examples

```
SigFilter()
SigFilter(logFC=2)
SigFilter(negLogFC=-1)
SigFilter(FDR=0.05)

esf <- EdgeSigFilter(logFC=2, FDR=0.05, logCPM=0)
LimmaSigFilter(logFC=1, FDR=0.05, aveExpr=10)
```

SigFilter-class *Base result filter for significantly regulated genes*

Description

Base result filter for significantly regulated genes

Slots

`posLogFC` Numeric, positive logFC threshold (larger values are kept)

`negLogFC` Numeric, negative logFC threshold (more negative values are kept)

`pValue` Numeric, p-value treshold (smaller values are kept)

`FDR` Numeric, FDR treshold

sigFilter<- *Replace the SigFilter of an CountDgeResult*

Description

Replace the SigFilter of an CountDgeResult

Usage

```
sigFilter(countDgeResult) <- value
```

Arguments

`countDgeResult`	An `EdgeResult` or `LimmaVoomResult` object
`value`	An SigFilter object

Value

An updated `countDgeResult` object

sigGene *Return significantly regulated genes*

Description

Return significantly regulated genes

Usage

```
sigGene(countDgeResult, contrast, value = NULL)

sigPosGene(countDgeResult, contrast, value = NULL)

sigNegGene(countDgeResult, contrast, value = NULL)
```

Arguments

countDgeResult	An EdgeResult object
contrast	Character, contrast(s) of interest
value	NULL or character string, if not NULL, it must be a column name in the feature annotation data.

Value

A vector of identifiers

Functions

- sigPosGene(): Only return positively significantly regulated genes
- sigNegGene(): Only return negatively significantly regulated genes

Examples

```
exMat <- matrix(rpois(120, 10), nrow=20, ncol=6)
exMat[2:4, 4:6] <- exMat[2:4, 4:6]+20
exMat[7:9, 1:3] <- exMat[7:9, 1:3]+20
exGroups <- gl(2,3, labels=c("Group1", "Group2"))
exDesign <- model.matrix(~0+exGroups)
colnames(exDesign) <- levels(exGroups)
exContrast <- matrix(c(-1,1), ncol=1, dimnames=list(c("Group1", "Group2"), c("Group2.vs.Group1")))
exDescon <- DesignContrast(exDesign, exContrast, groups=exGroups)
exFdata <- data.frame(GeneID=1:nrow(exMat),
  GeneSymbol=sprintf("Gene%d", 1:nrow(exMat)))
exPdata <- data.frame(Name=sprintf("Sample%d", 1:ncol(exMat)),
                      Group=exGroups)
exObj <- EdgeObject(exMat, exDescon,
                    fData=exFdata, pData=exPdata)
exDgeRes <- dgeWithEdgeR(exObj)
```

```
sigGenes(exDgeRes)
sigPosGenes(exDgeRes)
sigNegGenes(exDgeRes)
## specify the value type to return
sigGenes(exDgeRes, value="GeneSymbol")
sigPosGenes(exDgeRes, value="GeneSymbol")
sigNegGenes(exDgeRes, value="GeneSymbol")
```

`sigGeneBarchart` *Barchart of significantly regulated genes*

Description

Barchart of significantly regulated genes

Usage

```
sigGeneBarchart(
  edgeResult,
  logy = FALSE,
  scales = list(x = list(rot = 45), y = list(alternating = 1, tck = c(1, 0))),
  stack = FALSE,
  ylab = "Significant DEGs",
  col = c(positive = "orange", negative = "lightblue"),
  auto.key = list(columns = 2),
  ...
)
```

Arguments

`edgeResult`	An EdgeResult object
`logy`	Logical, whether y-axis should be log-10 transformed
`scales`	passed to `lattice::barchart`
`stack`	passed to `lattice::barchart`
`ylab`	passed to `lattice::barchart`
`col`	passed to `lattice::barchart`
`auto.key`	passed to `lattice::barchart`
`...`	passed to `lattice::barchart`

Value

A `trellis` object (lattice barchart).

sigGeneCounts *Return counts of significantly regulated genes*

Description

Return counts of significantly regulated genes

Usage

```
sigGeneCounts(countDgeResult, value = NULL)
```

Arguments

countDgeResult	An EdgeResult object
value	NULL or character string, if not NULL, it must be a column name in the feature annotation data.

Value

A data.frame containing counts of positively and negatively regulated genes, the sum, as well as total number of features

Examples

```
exMat <- matrix(rpois(120, 10), nrow=20, ncol=6)
exMat[2:4, 4:6] <- exMat[2:4, 4:6]+20
exMat[7:9, 1:3] <- exMat[7:9, 1:3]+20
exGroups <- gl(2,3, labels=c("Group1", "Group2"))
exDesign <- model.matrix(~0+exGroups)
colnames(exDesign) <- levels(exGroups)
exContrast <- matrix(c(-1,1), ncol=1, dimnames=list(c("Group1", "Group2"), c("Group2.vs.Group1")))
exDescon <- DesignContrast(exDesign, exContrast, groups=exGroups)
exFdata <- data.frame(GeneSymbol=sprintf("Gene%d", 1:nrow(exMat)))
exPdata <- data.frame(Name=sprintf("Sample%d", 1:ncol(exMat)),
                      Group=exGroups)
exObj <- EdgeObject(exMat, exDescon,
                    fData=exFdata, pData=exPdata)
exDgeRes <- dgeWithEdgeR(exObj)
sigGeneCounts(exDgeRes)
```

sigGeneDgeTable *Return dgeTable containing significantly regulated genes in respective contrasts*

Description

Return dgeTable containing significantly regulated genes in respective contrasts

Usage

```
sigGeneDgeTable(countDgeResult, value = "FeatrueName")
```

Arguments

countDgeResult	An EdgeResult object
value	A character string, it must be a column name in the feature annotation data. Default: FeatureName.

Value

A data.frame containing dgeTable of positively and negatively regulated genes in respective contrasts

Examples

```
exMat <- matrix(rpois(120, 10), nrow=20, ncol=6)
exMat[2:4, 4:6] <- exMat[2:4, 4:6]+20
exMat[7:9, 1:3] <- exMat[7:9, 1:3]+20
exGroups <- gl(2,3, labels=c("Group1", "Group2"))
exDesign <- model.matrix(~0+exGroups)
colnames(exDesign) <- levels(exGroups)
exContrast <- matrix(c(-1,1), ncol=1, dimnames=list(c("Group1", "Group2"), c("Group2.vs.Group1")))
exDescon <- DesignContrast(exDesign, exContrast, groups=exGroups)
exFdata <- data.frame(GeneSymbol=sprintf("Gene%d", 1:nrow(exMat)))
exPdata <- data.frame(Name=sprintf("Sample%d", 1:ncol(exMat)),
                      Group=exGroups)
exObj <- EdgeObject(exMat, exDescon,
                    fData=exFdata, pData=exPdata)
exDgeRes <- dgeWithEdgeR(exObj)
sigGeneDgeTable(exDgeRes, value="GeneSymbol")
```

sigGeneIdentifiers *Return gene identifiers of significant DGEs*

Description

Return gene identifiers of significant DGEs

Usage

```
sigGeneIdentifiers(dgeResult, contrast, sigFunc = isSig, value = NULL)
```

Arguments

dgeResult	An DgeResult object.
contrast	A character string, a contrast of interest.
sigFunc	A function, defining the type of significant genes.
value	NULL or character string, if not NULL, it must be a column name in the feature annotation data.

Value

A vector of character strings indicating the gene identifiers that are significantly regulated. If no defined types are found, either rownames or the first column is returned

See Also

geneIdentifierTypes

sigGenes *Return significantly regulated genes of all contrasts*

Description

Return significantly regulated genes of all contrasts

Usage

```
sigGenes(countDgeResult, value = NULL)

sigPosGenes(countDgeResult, value = NULL)

sigNegGenes(countDgeResult, value = NULL)
```

Arguments

countDgeResult	An EdgeResult object
value	NULL or character string, if not NULL, it must be a column name in the feature annotation data.

Value

A list of vectors of identifiers

Functions

- sigPosGenes(): Only return significantly positively regulated genes
- sigNegGenes(): Only return significantly negatively regulated genes

Note

TODO fix: add InputFeature

---

slurmEdgeR *Send an edgeR analysis job to SLURM*

Description

Send an edgeR analysis job to SLURM

Usage

```
slurmEdgeR(
  dgeList,
  designContrast,
  outdir = "edgeR_output",
  outfilePrefix = "an-unnamed-project-",
  overwrite = c("ask", "overwrite", "append", "no"),
  mps = FALSE,
  limmaVoom = FALSE,
  appendGmt = NULL,
  qos = c("3h", "1d", "3d", "15d", "interactive", "preempt"),
  rootPath = "/apps/rocs/pRED/groups/bioinfo/geneexpression",
  debug = FALSE
)
```

Arguments

dgeList	An DGEList object with counts, genes, and samples
designContrast	The DesignContrast object to model the data
outdir	Output directory of the edgeR script. Default value "edgeR_output".
outfilePrefix	Prefix of the output files. It can include directories, e.g. "data/outfile-". In case of NULL, temporary files will be created.
overwrite	If ask, the user is asked before an existing output directory is overwritten. If yes, the job will start and an existing directory will be overwritten anyway. If no, and if an output directory is present, the job will not be started.
mps	Logical, whether molecular-phenotyping analysis is run.
limmaVoom	Logical, whether the limma-voom model is run instead of the edgeR model.
appendGmt	NULL or character string, path to an additional GMT file for gene-set analysis. The option is passed to slurmEdgeRcommand and then to edgeRcommand.
qos	Character, specifying Quality of Service of Slurm. Available values include short (recommended default, running time cannot exceed 3 hours), interactive (useful if you wish to get the results from an interactive session), and long (useful if the job is expected to run more than three hours.) using srun and the 'interaction' queue of jobs instead of using sbatch.
rootPath	Character string, the directory of geneexpression scripts, under which bin/ngsDge_edgeR.Rscript is found.
debug	Logical, if TRUE, the source code of Rscript is used instead of the installed version. The option is passed to edgeRcommand.

Value

A list of two items, command, the command line call, and output, the output of the SLURM command in bash

Note

Even if the output directory is empty, if overwrite is set to no (or if the user answers no), the job will not be started.

Examples

```
mat <- matrix(rnbinom(100, mu=5, size=2), ncol=10)
rownames(mat) <- sprintf("gene%d", 1:nrow(mat))
myFac <- gl(2,5, labels=c("Control", "Treatment"))
y <- edgeR::DGEList(counts=mat, group=myFac)
myDesign <- model.matrix(~myFac); colnames(myDesign) <- levels(myFac)
myContrast <- limma::makeContrasts(Treatment, levels=myDesign)
myDescon <- DesignContrast(myDesign, myContrast)
## \dontrun{
## slurmEdgeR(y, myDescon, outfilePrefix="test", outdir=tempdir())
## }
```

slurmEdgeRcommand *Return the SLURM command to run the edgeR script*

Description

Return the SLURM command to run the edgeR script

Usage

```
slurmEdgeRcommand(
  dgeList,
  designContrast,
  outdir = "edgeR_output",
  outfilePrefix = "an-unnamed-project-",
  mps = FALSE,
  limmaVoom = FALSE,
  appendGmt = NULL,
  qos = c("3h", "1d", "3d", "15d", "interactive", "preempt"),
  params = "",
  rootPath = "/apps/rocs/pRED/groups/bioinfo/geneexpression",
  debug = FALSE
)
```

Arguments

dgeList	An DGEList object with counts, genes, and samples
designContrast	The DesignContrast object to model the data
outdir	Output directory of the edgeR script. Default value "edgeR_output".
outfilePrefix	Prefix of the output files. It can include directories, e.g. "data/outfile-". In case of NULL, temporary files will be created.
mps	Logical, whether molecular-phenotyping analysis is run.
limmaVoom	Logical, whether the limma-voom model is run instead of the edgeR model
appendGmt	NULL or character string, path to an additional GMT file for gene-set analysis. The option is passed to edgeRcommand.
qos	Character, specifying Quality of Service of Slurm. Available values include short (recommended default, running time cannot exceed 3 hours), interactive (useful if you wish to get the results from an interactive session), and normal (useful if the job is expected to run more than three hours.) using srun and the 'interaction' queue of jobs instead of using sbatch.
params	Character, further parameters to pass to sbatch, for instance "–partition ANOTHER_PARITION"
rootPath	Character string, the directory of geneexpression scripts, under which bin/ngsDge_edgeR.Rscript is found.

debug — Logical, if TRUE, the source code of Rscript is used instead of the installed version. The option is passed to edgeRcommand.

This function wraps the function edgeRcommand to return the command needed to start a SLURM job.

It uses outdir to specify slurm output and error files as in the same directory of outdir. And the job name is set as the name of the output directory.

Value

A character string containing the SLURM sbatch command to submit the edgeR analysis job.

See Also

edgeRcommand

Examples

```
mat <- matrix(rnbinom(100, mu=5, size=2), ncol=10)
rownames(mat) <- sprintf("gene%d", 1:nrow(mat))
myFac <- gl(2,5, labels=c("Control", "Treatment"))
y <- edgeR::DGEList(counts=mat, group=myFac)
myDesign <- model.matrix(~myFac); colnames(myDesign) <- levels(myFac)
myContrast <- limma::makeContrasts(Treatment, levels=myDesign)
myDesCon <- DesignContrast(designMatrix=myDesign,
                           contrastMatrix=myContrast)
mytempdir <- tempdir()
slurmEdgeRcommand(y, myDesCon, outfilePrefix="test", outdir=mytempdir)
## clean up
unlink(file.path(mytempdir, "input_data"), recursive=TRUE)
slurmFile <- file.path(dirname(mytempdir),
    paste0("slurm-", basename(mytempdir), ".sh"))
unlink(slurmFile)
```

---

smearPlot *Smear plot*

Description

Smear plot

Usage

```
smearPlot(object, ...)

## S4 method for signature 'EdgeResult'
smearPlot(
  object,
  contrast = NULL,
```

```
  freeRelation = FALSE,
  xlab = "Average logCPM",
  ylab = "logFC",
  pch = 19,
  cex = 0.2,
  smearWidth = 0.5,
  panel.first = grid(),
  smooth.scatter = FALSE,
  lowess = FALSE,
  multipage = FALSE,
  ...
)
```

Arguments

object	An object
...	Other parameters
contrast	Character, contrast of interest
freeRelation	Logical
xlab	Character
ylab	Character
pch	Character or integer
cex	Numeric
smearWidth	Numeric
panel.first	Grid
smooth.scatter	Logical
lowess	Logical
multipage	Logical

Value

Called for its side effect of plotting; returns invisibly NULL.

Methods (by class)

- `smearPlot(EdgeResult)`: Method for EdgeResult

split.DGEList *Split a DGEList object by a factor of samples (default) or genes*

Description

Split a DGEList object by a factor of samples (default) or genes

Usage

```
## S3 method for class 'DGEList'
split(x, f, drop = FALSE, bySample = TRUE, sampleDropLevels = TRUE, ...)

splitDGEList(x, f, drop = FALSE, bySample = TRUE, sampleDropLevels = TRUE, ...)
```

Arguments

x	A DGEList object
f	A factor vector. Other types will be coereced into factors.
drop	Not used now
bySample	Logical, if TRUE, the samples are split. Otherwise, genes are split.
sampleDropLevels	Logical, if TRUE, unused levels in factors in the sample annotation are dropped
...	Not used so far.

Value

A DGEListList object, a list of DGEList objects split by the factor f.

Functions

- splitDGEList(): A wrapper of split.DGEList

Examples

```
y1 <- matrix(rnbinom(1000, mu=5, size=2), ncol=4)
genes1 <- data.frame(GeneSymbol=sprintf("Gene%d", 1:nrow(y1)),
  GeneType=gl(5,50))
rownames(y1) <- rownames(genes1) <- 1:nrow(y1)
anno1 <- data.frame(treatment=gl(2,2, labels=c("ctrl", "tmt")),
    donor=factor(rep(c(1,2), each=2)))
d1 <- DGEList(counts=y1, genes=genes1, samples=anno1)

d1SampleSplit <- split(d1, d1$samples$donor)
d1GeneSplit <- split(d1, d1$genes$GeneType, bySample=FALSE)
```

staticGeneLevelPlots *Make static gene-level plots of an EdgeResult object*

Description

Make static gene-level plots of an EdgeResult object

Usage

```
staticGeneLevelPlots(edgeResult)
```

Arguments

edgeResult	An EdgeResult object

Value

Called for its side effect of generating plots; returns invisibly NULL.

Examples

```
edgeObj <- exampleEdgeObject()
edgeRes <- dgeWithEdgeR(edgeObj)
staticGeneLevelPlots(edgeRes)

limmaVoomRes <- dgeWithEdgeR(edgeObj)
staticGeneLevelPlots(limmaVoomRes)
```

svaseqRemove *Detect surrogate variables from count data and remove their effects VSN-transformed counts*

Description

Detect surrogate variables from count data and remove their effects VSN-transformed counts

Usage

```
svaseqRemove(dgeList, design, nullModel, verbose = FALSE, offset)
```

Arguments

dgeList	An DGEList object
design	Design matrix
nullModel	Null model matrix
verbose	Logical
offset	If provided, it is passed to pcaScores.
	In case no significant surrogate variables are detected, PCA analysis is applied to the vsn-transformed matrix.

Value

A list with following items

sva Results of svaseq

vsnFit Fit object of vsn

vsnMat Fitted matrix of vsn

vsnBatchRemoved Fitted matrix of vsn, with surrogates' effect removed

vsnBatchRemovedPca PCA object derived from vsnBatchRemoved

vsnBatchRemovedPcaScores PCA scores with annotations

designWithSV Design matrix with surrogates variables appended if any

Note

This function needs to be harmonized with the other SVA functions. The reason is that svaseq was for a long time not stable until recently. Therefore this function is written later, and unfortunately the outcome is not harmonized yet.

See Also

pcaScores

Examples

```
y1org <- matrix(rnbinom(4000, mu=5, size=2), ncol=8)
genes1 <- data.frame(GeneSymbol=sprintf("Gene%d", 1:nrow(y1org)))
y1 <- y1org
y1[30:120, 4:7] <- y1[30:120, 4:7]+9 ## mimicking batch effect
rownames(y1org) <- rownames(y1) <- rownames(genes1) <- 1:nrow(y1)
anno1 <- data.frame(treatment=gl(2,4, labels=c("ctrl", "tmt")),
    donor=factor(rep(c(1,2), 4)))
d1 <- DGEList(counts=y1, genes=genes1, samples=anno1)
d2 <- DGEList(counts=y1org, genes=genes1, samples=anno1)

design <- model.matrix(~treatment+donor, data=d1$samples)
nullModel <-  model.matrix(~donor, data=d1$samples)
d1VsnSvaRes <- svaseqRemove(d1, design, nullModel)
d2VsnSvaRes <- svaseqRemove(d2, design, nullModel)
```

tagwiseBCV *Tagwise biological coefficients of variance*

Description

Tagwise biological coefficients of variance

Usage

```
tagwiseBCV(x)

## S4 method for signature 'DGEList'
tagwiseBCV(x)

## S4 method for signature 'EdgeResult'
tagwiseBCV(x)
```

Arguments

x	An object

Value

A numeric vector of tagwise BCV values.

Methods (by class)

- `tagwiseBCV(DGEList)`: method for DGEList
- `tagwiseBCV(EdgeResult)`: method for EdgeResult

testGLM *Test GLM*

Description

Test GLM

Usage

```
testGLM(object, fit)

## S4 method for signature 'EdgeObject,DGEGLM'
testGLM(object, fit)
```

Arguments

object	An object
fit	A fit object

Value

An EdgeResult object containing the test results.

Methods (by class)

- testGLM(object = EdgeObject, fit = DGEGLM): Method for EdgeObject and DGEGLM.

---

topDgeExpression *Return raw expression of top differentially expressed genes of multiple contrasts*

Description

Return raw expression of top differentially expressed genes of multiple contrasts

Usage

```
topDgeExpression(
  edgeResult,
  ntop = 10,
  contrast = NULL,
  exprsFun = function(dgeList) cpm(dgeList, log = TRUE)
)
```

Arguments

edgeResult	An EdgeResult object.
ntop	Integer, number of top differentially expressed genes.
contrast	NULL, which means all contrasts are considered, or a vector of character strings or of integer values, which specify which contrasts are considered.
exprsFun	A function to derive expression values from EdgeResult.

Value

A tibble object with Contrast in the first column and a wide table of raw expression, feature annotation, and sample annotations in the rest columns

See Also

topDgeExpressionByContrast

topDgeExpressionByContrast *Return raw expression of top differentially expressed genes of one contrast*

Description

Return raw expression of top differentially expressed genes of one contrast

Usage

```
topDgeExpressionByContrast(
  edgeResult,
  ntop = 10,
  contrast,
  exprsFun = function(dgeList) cpm(dgeList, log = TRUE)
)
```

Arguments

edgeResult	An EdgeResult object.
ntop	Integer, number of top differentially expressed genes.
contrast	A character string or an integer value, specifying which contrast is considered.
exprsFun	A function to derive expression values from EdgeResult.

Value

A tibble object with Contrast in the first column and a wide table of raw expression, feature annotation, and sample annotations in the rest columns

tpm *Convert count matrix to TPM values*

Description

Convert count matrix to TPM values

Usage

```
tpm(x, gene.length)
```

Arguments

x	A count matrix or other objects that can be converted to a matrix by as.matrix
gene.length	A numeric vector of the same length as the row count of x, giving gene lengths

Value

transcripts per million (TPM) values

See Also

rpkm2tpm

Examples

```
testMatrix <- matrix(rnbinom(200, size=5, prob=0.1), nrow=20, ncol=10)
testMatrixGeneLen <- as.integer(10^rnorm(20, mean=3, sd=0.5))
testMatrixTpm <- tpm(testMatrix, testMatrixGeneLen)
```

---

trendedBCV *Trended biological coefficients of variance*

Description

Trended biological coefficients of variance

Usage

```
trendedBCV(x)

## S4 method for signature 'DGEList'
trendedBCV(x)

## S4 method for signature 'EdgeResult'
trendedBCV(x)
```

Arguments

x	An object

Value

A numeric vector of trended BCV values.

Methods (by class)

- `trendedBCV(DGEList)`: method for DGEList
- `trendedBCV(EdgeResult)`: method for EdgeResult

updateContrastMatrixWithSV

Update a contrast matrix given a surrogate variable matrix

Description

Update a contrast matrix given a surrogate variable matrix

Usage

```
updateContrastMatrixWithSV(contrastMatrix, svMatrix)
```

Arguments

contrastMatrix	A contrast matrix
svMatrix	A surrogate-variable matrix, for instance returned by sva

Value

An updated matrix, with surrogate matrix variables appended to the rows

Examples

```
exCounts <- matrix(rpois(12000, 10), nrow=2000, ncol=6)
exCounts[1:100, 2:3] <- exCounts[1:100,2:3]+20
exDesign <- model.matrix(~gl(2,3))
colnames(exDesign) <- c("Baseline", "Treatment")
exContrast <- limma::makeContrasts("Treatment"="Treatment", levels=exDesign)
exVoomSvaRes <- voomSVA(exCounts, exDesign)
updateContrastMatrixWithSV(exContrast, exVoomSvaRes)
```

updateDesignMatrixBySVA

Update design matrix by SVA

Description

Update design matrix by SVA

Usage

```
updateDesignMatrixBySVA(object, design, ...)

## S4 method for signature 'DGEList,formula'
updateDesignMatrixBySVA(object, design, ...)
```

Arguments

object	An object
design	Design matrix or formula
...	Other parameters

Value

A design matrix updated with surrogate variables.

Methods (by class)

- updateDesignMatrixBySVA(object = DGEList, design = formula): for DGEList and formula

---

updateSigFilter *Update the SigFilter*

Description

Update the SigFilter

Usage

```
updateSigFilter(countDgeResult, logFC, posLogFC, negLogFC, pValue, FDR, ...)
```

Arguments

countDgeResult	An EdgeResult or LimmaVoomResult object
logFC	Numeric
posLogFC	Numeric
negLogFC	Numeric
pValue	Numeric
FDR	Numeric
...	Other parameters, now used ones including aveExpr (for LimmaSigFilter) and logCPM (for EdgeSigFilter)

Value

An updated CountDgeResult object with updated SigFilter

volcanoPlot *Volcano plot*

Description

Volcano plot

Usage

```
volcanoPlot(object, ...)

## S4 method for signature 'EdgeResult'
volcanoPlot(
  object,
  contrast = NULL,
  freeRelation = FALSE,
  colramp = ribiosPlot::heat,
  multipage = FALSE,
  yValue = c("PValue", "FDR"),
  xlim = NULL,
  ylim = NULL,
  main = NULL,
  topLabel = NULL,
  labelType = NULL,
  ...
)
```

Arguments

object	An object
...	Other parameters
contrast	Character, contrast of interest. If NULL, all contrasts are used
freeRelation	Logical.
colramp	Function, color palette.
multipage	Logical.
yValue	Character string, either PValue or FDR.
xlim	NULL or a numeric vector of two
ylim	NULL or a numeric vector of two.
main	Character, title.
topLabel	NULL or an integer number, number of top features to be labelled
labelType	NULL or a character string, a column name in the feature annotation

Value

Called for its side effect of plotting; returns invisibly NULL.

Methods (by class)

- `volcanoPlot(EdgeResult)`: Method for EdgeResult

`voom` *Perform VOOM analysis*

Description

Perform VOOM analysis

Usage

```
voom(object, ...)

## S4 method for signature 'DGEList'
voom(object, ...)

## S4 method for signature 'matrix'
voom(object, ...)

## S4 method for signature 'ExpressionSet'
voom(object, ...)

## S4 method for signature 'EdgeObject'
voom(object, ...)
```

Arguments

`object`	An object
`...`	Other parameters

Value

An `EList` object containing voom-transformed values.

Methods (by class)

- `voom(DGEList)`: Method for DGEList
- `voom(matrix)`: Method for matrix
- `voom(ExpressionSet)`: Method for matrix
- `voom(EdgeObject)`: Method for EdgeObject, norm.factors are calculated first if not done yet

voomLimma *Perform the voom+limma procedure*

Description

Perform the voom+limma procedure

Usage

```
voomLimma(
  dgeList,
  design,
  contrasts,
  normalize.method = "none",
  block = NULL,
  correlation = NULL,
  weights = NULL,
  plot = FALSE,
  ...
)
```

Arguments

dgeList	A DGEList object, it should be ideally already filtered
design	The design matrix
contrasts	The contrast matrix
normalize.method	Character string, passed to voom, keep it none unless you are sure
block	Blocking factor, passed to voom
correlation	Correlation between duplicates, passed to voom
weights	Weights, passed to voom
plot	Logical, whether the variance-mean relationship should be ploted
...	Passed to eBayes

Value

MArrayLM object returned by eBayes, with voom object in the voom element of the list

Examples

```
y <- matrix(rnbinom(10000,mu=5,size=2),ncol=4)
d <- edgeR::DGEList(counts=y, group=rep(1:2,each=2))
d <- edgeR::calcNormFactors(d)
design <- model.matrix(~gl(2,2))
colnames(design) <- c("baseline", "treatment")
contrasts <- limma::makeContrasts("treatment", levels=design)
```

```
dvl <- voomLimma(d, design=design, contrasts=contrasts)
```

---

voomRemoveSV *Apply SVA to voom-transformed count data, and return the voom expression matrix with surrogate variables' effect removed*

Description

Apply SVA to voom-transformed count data, and return the voom expression matrix with surrogate variables' effect removed

Usage

```
voomRemoveSV(counts, designMatrix)
```

Arguments

counts	A matrix of counts
designMatrix	Design matrix

Value

The voom expression matrix, with SV effects removed

A numeric matrix of voom-transformed expression values with surrogate variable effects removed.

Examples

```
exCounts <- matrix(rpois(12000, 10), nrow=2000, ncol=6)
exCounts[1:100, 2:3] <- exCounts[1:100,2:3]+20
exDesign <- model.matrix(~gl(2,3))
head(voomRemoveSV(exCounts, designMatrix=exDesign))
## compare the results without SV removal, note the values in the
## second and third column are much larger than the rest
head(voom(exCounts, exDesign)$E)
```

voomSVA *Run SVA on a count matrix transformed by voom*

Description

Run SVA on a count matrix transformed by voom

Usage

```
voomSVA(object, design, ...)

## S4 method for signature 'matrix,matrix'
voomSVA(object, design)

## S4 method for signature 'DGEList,matrix'
voomSVA(object, design)

## S4 method for signature 'DGEList,formula'
voomSVA(object, design)
```

Arguments

object	A count matrix
design	Design matrix or formula
...	Other parameters

Value

SV matrix

Methods (by class)

- voomSVA(object = matrix, design = matrix): Method for count matrix and design matrix
- voomSVA(object = DGEList, design = matrix): Method for DGEList and design matrix
- voomSVA(object = DGEList, design = formula): Method for count matrix and design formula

Examples

```
set.seed(1887)
exCounts <- matrix(rpois(12000, 10), nrow=2000, ncol=6)
exCounts[1:100, 2:3] <- exCounts[1:100,2:3]+20
exDesign <- model.matrix(~gl(2,3))
voomSVA(exCounts, design=exDesign)
```

writeBiokitSampleAnnotation *Write sample annotation into a tab-delimited file to start the Biokit pipeline*

Description

Write sample annotation into a tab-delimited file to start the Biokit pipeline

Usage

```
writeBiokitSampleAnnotation(df, con)
```

Arguments

df	A data.frame or anything that can be converted to a data.frame
con	Connection, can be a character string indicating file name

Value

Called for its side effect of writing the sample annotation file; returns invisibly NULL.

Note

Starting from version 1.0-36, the function checks the input `data.frame` or `tbl_df` before writing to the file

Examples

```
testDf <- data.frame(ID=LETTERS[1:4],
   GROUP=gl(2,2),
   FASTQ1=sprintf("%s_R1.fastq.gz", LETTERS[1:4]),
   FASTQ2=sprintf("%s_R1.fastq.gz", LETTERS[1:4]))
tmp <- tempfile()
writeBiokitSampleAnnotation(testDf, con=tmp)
readLines(tmp)
```

`writeDGEList` *Write an DGEList object as plain files for downstream analysis*

Description

Write an DGEList object as plain files for downstream analysis

Usage

```
writeDGEList(
  dgeList,
  exprs.file,
  fData.file,
  pData.file,
  group.file,
  groupLevels.file,
  feat.name = NULL,
  feat.desc = NULL
)
```

Arguments

`dgeList`	An DGEList object
`exprs.file`	File name where counts are saved
`fData.file`	File name where feature annotations are saved
`pData.file`	File name where sample annotations are saved
`group.file`	File name where the sample group information is saved
`groupLevels.file`	File where the sample group levels are saved
`feat.name`	Feature names. Can be a column name in `genes` of the DGEList object, or a vector of the same length as the fetaures. If `NULL`, row names of the count matrix are used.
`feat.desc`	Feature descriptions, used in GCT files. If `NULL`, 'GeneSymbol' will be used if the column is present, otherwise no description will be used
	Expression values are saved by default in the gct format, unless the file name ends with tsv in which case a tab-separated value (TSV) file will be saved.
	Sample group and group level information are derived from the `group` column of the sample annotation.

Value

Called for its side effect of writing files; returns invisibly NULL.

Note

In case the input matrix has no feature name, the feature names are set to be the integer array starting from 1.

In case no genes item is available in the DGEList, a minimal data.frame containing one column, Feature, is exported with row names of the count matrix used as both row names as well as the content of the Feature column.

Examples

```
y <- matrix(rnbinom(10000,mu=5,size=2),ncol=4)
d <- DGEList(counts=y, group=rep(1:2,each=2))

exprsFile <- tempfile()
fDataFile <- tempfile()
pDataFile <- tempfile()
groupFile <- tempfile()
groupLevelsFile <- tempfile()
writeDGEList(d, exprs.file=exprsFile, fData.file=fDataFile, pData.file=pDataFile,
  group.file=groupFile, groupLevels.file=groupLevelsFile)

head(ribiosIO::read_gct_matrix(exprsFile))
head(ribiosIO::readMatrix(fDataFile))
head(ribiosIO::readMatrix(pDataFile))
head(readLines(groupFile))
head(readLines(groupLevelsFile))
```

---

writeDgeTables *Write DGE tables in individual files, and the merged table in one file*

Description

Write DGE tables in individual files, and the merged table in one file

Usage

```
writeDgeTables(edgeResult, outdir = getwd())
```

Arguments

edgeResult	An EdgeResult object
outdir	Output directory

Value

NULL, side effects are used

writeDgeTablesWithPseudoT

Write dgeTables with pseudo T statistics

Description

Write dgeTables with pseudo T statistics

Usage

```
writeDgeTablesWithPseudoT(edgeResult, outdir = getwd())
```

Arguments

edgeResult	An EdgeResult object
outdir	Output directory

Value

NULL, side effects are used

See Also

dgeTableWithPseudoT

writeTruncatedDgeTables

Write truncated DGE tables

Description

Write truncated DGE tables

Usage

```
writeTruncatedDgeTables(edgeResult, outdir = getwd())
```

Arguments

edgeResult	An EdgeResult object
outdir	Output directory

Value

NULL, side effects are used

Index