Package: ribiosGSEA (via r-universe)

May 27, 2026

Type Package

Title Gene-Set Enrichment Analysis Tools in 'ribios'

Version 1.6.7

Date 2026-03-28

Description Provides data structure and functions for gene-set analysis and post-processing of analysis results.

Depends R (>= 4.1.0)

Imports Rcpp, methods, ribiosIO, ribiosUtils(>= 1.1-9), ribiosExpression, ribiosPlot, ribiosNGS, XML, parallel, limma, Biobase, BioQC (>= 1.13.2), edgeR, dplyr, data.table, httr, jsonlite, gage, readr, graph, igraph

Suggests testthat, knitr, rmarkdown

License GPL-3

Collate 'ribiosGSEA-package.R' 'AllClasses.R' 'AllGenerics.R' 'AllMethods.R' 'AnnoBroadGseaResItem.R' 'GeMS.R' 'RcppExports.R' 'utils.R' 'biosCamera.R' 'buildBroadGSEAcomm.R' 'gse.R' 'doGse.R' 'fingerprint.R' 'fisher.R' 'fishersMethod.R' 'geneSetPerm.R' 'gmt.R' 'gseaScores.R' 'kendallW.R' 'kmeansGenesets.R' 'list2mat.R' 'parseBroadGSEA.R' 'parseTopTable.R' 'post-processing.R' 'prettyRonetGenesetNames.R' 'strWithNumber.R'

LinkingTo Rcpp

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.3

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

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

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

Remotes github::bedapub/ribiosIO, github::bedapub/ribiosPlot, github::bedapub/ribiosExpression, github::bedapub/ribiosNGS

Config/pak/sysreqs cmake libglpk-dev 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 2026-03-28 16:45:17 UTC

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

RemoteRef HEAD

RemoteSha 78ebff2a630b26a851ac5ea3e3a64eda6e0c9a2d

Contents

`[,AnnoBroadGseaRes,ANY,ANY,ANY-method`

Subset an AnnoBroadGseaRes object

Description

Subset an AnnoBroadGseaRes object

Usage

```
## S4 method for signature 'AnnoBroadGseaRes,ANY,ANY,ANY'
x[i, j, ..., drop = FALSE]
```

Arguments

x	An AnnoBroadGseaRes object
i	An integer or logical subsetting index
j	Not used
...	Not used
drop	Not used

Value

A subset of the original data as an AnnoBroadGseaRes object

[,FisherResultList,ANY,missing,missing-method

Subset a FisherResultList object by indexing

Description

Subset a FisherResultList object by indexing

Usage

```
## S4 method for signature 'FisherResultList,ANY,missing,missing'
x[i, j, ..., drop = FALSE]
```

Arguments

x	A FisherResultList object
i	An integer or logical subsetting index
j	Not used
...	Not used
drop	Not used

Value

A subset of the original data as an FisherResultList object

[,FisherResultList,character,character,missing-method *Subset a FisherResultList object by namespace and name*

Description

Subset a FisherResultList object by namespace and name

Usage

```
## S4 method for signature 'FisherResultList,character,character,missing'
x[i, j, ..., drop = TRUE]
```

Arguments

x	A FisherResultList object
i	Character string, gene-set namespace
j	Character string, gene-set name
...	Not used
drop	Not used

AnnoBroadGseaRes *Convert a list of AnnoBroadGseaResItem objects to a list*

Description

Convert a list of AnnoBroadGseaResItem objects to a list

Usage

```
AnnoBroadGseaRes(object)
```

Arguments

object	A list of AnnoBroadGseaResItem

Value

An AnnoBroadGseaRes object

AnnoBroadGseaRes-class *Annotated BROAD GSEA Results for one contrast*

Description

Annotated BROAD GSEA Results for one contrast

Value

An object of class AnnoBroadGseaRes.

AnnoBroadGseaResItem *Convert a BroadGseaResItem object to an AnnoBroadGseaResItem object*

Description

Convert a BroadGseaResItem object to an AnnoBroadGseaResItem object

Usage

```
AnnoBroadGseaResItem(object, genes, geneValues)
```

Arguments

object	A BroadGseaResItem object
genes	A character string vector
geneValues	A numeric vector

Value

An annoBroadGseaResItem object

AnnoBroadGseaResItem-class
Annotated BROAD GSEA result item

Description

Annotated BROAD GSEA result item

Value

An object of class `AnnoBroadGseaResItem`.

Slots

`gsGenes` Vector of character strings, gene-set genes

`gsGeneValues` Vector of numeric values, statistics of gene-set genes

AnnoBroadGseaResList-class
A list of AnnoBroadGseaRes objects

Description

A list of AnnoBroadGseaRes objects

Value

An object of class `AnnoBroadGseaResList`.

as.data.frame,FisherResultList-method
Convert an FisherResultList object into a data.frame

Description

Convert an FisherResultList object into a data.frame

Usage

```
## S4 method for signature 'FisherResultList'
as.data.frame(x, row.names = NULL)
```

Arguments

x	An FisherResultList object
row.names	Character strings.

Value

A data.frame

---

biosCamera *An adapted and enhanced version of limma::camera*

Description

An adapted and enhanced version of limma::camera

Usage

```
biosCamera(
  y,
  index,
  design = NULL,
  contrast = ncol(design),
  weights = NULL,
  geneLabels = NULL,
  use.ranks = FALSE,
  allow.neg.cor = FALSE,
  trend.var = FALSE,
  sort = FALSE,
  .fixed.inter.gene.cor = NULL,
  .approx.zscoreT = FALSE
)
```

Arguments

y	a numeric matrix of log-expression values or log-ratios of expression values, or any data object containing such a matrix. Rows correspond to probes and columns to samples. Any type of object that can be processed by getEAWP is acceptable.
index	an index vector or a list of index vectors. Can be any vector such that y[index,] of statistic[index] selects the rows corresponding to the test set. The list can be made using ids2indices.
design	Design matrix
contrast	contrast of the linear model coefficients for which the test is required. Can be an integer specifying a column of design, or else a numeric vector of same length as the number of columns of design.

`weights` numeric matrix of observation weights of same size as `y`, or a numeric vector of array weights with length equal to `ncol(y)`, or a numeric vector of gene weights with length equal to `nrow(y)`.

`geneLabels` Labels of the features in the input matrix.

`use.ranks` do a rank-based test (TRUE) or a parametric test (FALSE)?

`allow.neg.cor` should reduced variance inflation factors be allowed for negative correlations?

`trend.var` logical, should an empirical Bayes trend be estimated? See `eBayes` for details.

`sort` logical, should the results be sorted by p-value?

`.fixed.inter.gene.cor`
Numeric value, vector, or `NULL`/`NA`, advanced parameter corresponding to `inter.gene.cor` in the original implementation in limma. If set, gene-sets are set to have the fixed inter-gene correlation; the vector will be recycled to meet the correct length. If set as `NULL`/`NA`, correlations are estimated from each gene-set.

`.approx.zscoreT`
logical, advanced parameter only used for debugging purposes. If `TRUE`, the code is expected to return the exact same results as edgeR::camera (version 3.20.9), and maybe faster in execution.

The function was adapted from `camera`, with following improvments

1. The output data.frame is more user-friendly
2. The column 'FDR' is always present, even when only one gene-set was tested
3. Scores are calculated, defined as `log10(pValue)*I(directionality)`, where `I(directionality)` equals `1` if the directionality is `Up` and `-1` if the directionality is `Down`
4. Contributing genes and statistics are printed

Value

A `data.frame` with one row per set and the following columns:

GeneSet Gene set name

NGenes Number of genes in the set

Correlation Estimated correlation

EffectSize Estimated difference between the mean values of genes in the geneset and the background genes

Direction Direction of set-wise regulation, `Up` or `Down`

Score Gene-set enrichment score, defined as `log10(pValue)*I(directionality)`, where `I(directionality)` equals `1` if the directionality is `Up` and `-1` if the directionality is `Down`

ContribuingGenes A character string, containing all genes labels of genes that are in the set and regulated in the same direction as the set-wise direction, and the respective statistic

Note

Since limma 3.29.6, the default setting of allow.neg.cor changes from TRUE to FALSE, and a new parameter, inter.gene.cor, is added with the default value of 0.01, namely a prior inter-gene correlation is set for all gene sets. Currently, `biosCamera` does not have the parameter `inter.gene.cor`, but `allow.neg.cor` is set by default to `FALSE` to be consistent with the latest camera function.

Examples

```
y <- matrix(rnorm(1000*6),1000,6)
design <- cbind(Intercept=1,Group=c(0,0,0,1,1,1))
# First set of 20 genes are genuinely deferentially expressed
index1 <- 1:20
y[index1,4:6] <- y[index1,4:6]+1
# The second set of 20 genes are not
index2 <- 21:40
biosCamera(y, index1, design)
biosCamera(y, index2, design)
biosCamera(y, list(index1, index2), design)

# compare with the output of camera: columns 'GeneSet', 'Score',
# 'ContributingGenes' are missing, and in case \code{inter.gene.cor} is (as
# default) set to a numeric value, the column 'Correlation' is also missing

limmaDefOut <- limma::camera(y, index1, design)
limmaCorDefOut <-
    limma::camera(y, index1, design, inter.gene.cor=NA)

## Not run:
  # when \code{.approx.zscoreT=TRUE},  PValue reported by
  # \code{limma::camera(inter.gene.cor=NA)} and \code{ribiosGSEA::biosCamera}
  # should equal
  biosCorOut <- biosCamera(y, index1, design, .approx.zscoreT=TRUE)

  # when \code{.fixed.inter.gene.cor=0.01} and \code{.approx.zscoreT=TRUE},
  # PValue reported by \code{limma::camera} and \code{ribiosGSEA::biosCamera}
  # should equal
  biosFixCorOut <- biosCamera(y, index1, design,
      .fixed.inter.gene.cor=0.01, .approx.zscoreT=TRUE)
  testthat::expect_equal(biosFixCorOut$PValue, limmaDefOut$PValue)
  testthat::expect_equal(biosCorOut$PValue, limmaCorDefOut$PValue)

## End(Not run)
```

`BroadGseaResItem-class` *A S4 class representing the atom structure of results of the BROAD GSEA tool*

Description

A S4 class representing the atom structure of results of the BROAD GSEA tool

Value

An object of class `BroadGseaResItem`.

Slots

`geneset` Character, gene-set name

`es` Numeric, enrichment score

`nes` Numeric, normalised enrichment score

`np` Numeric

`fdr` Numeric, false discovery rate

`fwer` Numeric, family-wise error rate

`geneIndices` Integer vector, gene indices

`esProfile` Numeric, enrichment score profile

`coreEnrichThr` Numeric

`buildBroadGSEAcomm` *Build the command-line command to run BROAD GSEA*

Description

Build the command-line command to run BROAD GSEA

Usage

```
buildBroadGSEAcomm(
  gseaJar,
  javaBin,
  rnkFiles,
  gmtFile,
  chipFile,
  nperm = 1000L,
  collapse = FALSE,
  plotTopX = 25,
  outdir = "./",
  addShebang = TRUE
)
```

Arguments

`gseaJar`	Character string, full file name of BROAD GSEA (gene permutation) jar file
`javaBin`	Character string, java binary file
`rnkFiles`	Character string, rank files
`gmtFile`	A `GMT` file encoding GMT files to be used
`chipFile`	A `CHIP` file encoding feature annotation
`nperm`	Integer, number of permutations
`collapse`	Logical, whether to collapse duplicated features

plotTopX	Integer, top gene-sets to be visualized
outdir	Character string, the path of output
addShebang	Logical, whether to add Shebang to the script
	The command builds command-line command to run gene-permutation GSEA over many rank files.

Value

A character vector of shell commands to run GSEA.

`camera.EdgeResult` *Run CAMERA method using EdgeResult*

Description

Run CAMERA method using EdgeResult

Usage

```
## S3 method for class 'EdgeResult'
camera(y, gmtList, doParallel = FALSE, ...)
```

Arguments

y	A EdgeResult object
gmtList	Gene set collections, for example read by `readGmt`, with namespace.
doParallel	Logical, whether `parallel::mclapply` should be used. Since at the current setting it makes a job running forever, use `TRUE` only if you are debugging the code.
...	Not used
	Note that the EdgeResult object must have a column 'GeneSymbol' in its `fData`.

Value

A `data.frame` containing CAMERA results.

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(-1,1,1,-1), ncol=2, dimnames=list(c("Group1", "Group2"),
  c("Group2.vs.Group1", "Group1.vs.Group2")))
exDescon <- DesignContrast(exDesign, exContrast, groups=exGroups)
exFdata <- data.frame(GeneSymbol=sprintf("GeneSymbol%d", 1:nrow(exMat)))
exPdata <- data.frame(Name=sprintf("Sample%d", 1:ncol(exMat)),
```

```
                    Group=exGroups)
exDgeList <- DGEList(exMat, genes=exFdata, samples=exPdata)
exDgeList <- edgeR::estimateDisp(exDgeList, exDesign)
exEdgeObject <- EdgeObject(exDgeList, exDescon)
exEdgeRes <- ribiosNGS::dgeWithEdgeR(exEdgeObject)
exGmt <- BioQC::GmtList(list(GeneSet1=sprintf("GeneSymbol%d", 1:5),
  GeneSet2=sprintf("GeneSymbol%d", 6:10)))

exCameraRes <- camera(exEdgeRes, exGmt)
```

`camera.LimmaVoomResult` *Run the CAMERA method using LimmaVoomResult*

Description

Run the CAMERA method using LimmaVoomResult

Usage

```
## S3 method for class 'LimmaVoomResult'
camera(y, gmtList, doParallel = FALSE, ...)
```

Arguments

`y`	A LimmaVoomResult object
`gmtList`	Gene set collections, for example read by `readGmt`
`doParallel`	Logical, whether `parallel::mclapply` should be used. Since at the current setting it makes a job running forever, use `TRUE` only if you are debugging the code.
`...`	Passed to `cameraLimmaVoomResultsByContrast` Note that the LimmaVoomResult object must have a column 'GeneSymbol' in its `fData`.

Value

A `data.frame` containing CAMERA results.

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(GeneSymbol=sprintf("Gene%d", 1:nrow(exMat)))
```

```
exPdata <- data.frame(Name=sprintf("Sample%d", 1:ncol(exMat)),
                      Group=exGroups)
exDgeList <- DGEList(exMat, genes=exFdata, samples=exPdata)
exDgeList <- edgeR::estimateDisp(exDgeList, exDesign)
edgeObj <- EdgeObject(exDgeList, exDescon)
limmaVoomRes <- ribiosNGS::dgeWithLimmaVoom(edgeObj)
exGmt <- BioQC::GmtList(list(GeneSet1=sprintf("GeneSymbol%d", 1:5),
  GeneSet2=sprintf("GeneSymbol%d", 6:10)))

camera(limmaVoomRes, exGmt)
```

`cameraDGEListByContrast` *Apply the CAMERA method to a DGEList object and a contrast*

Description

Apply the CAMERA method to a DGEList object and a contrast

Usage

```
cameraDGEListByContrast(dgeList, index, design, contrasts, doParallel = FALSE)
```

Arguments

`dgeList`	A DGEList object, with GeneSymbol available, and dispersion must be estimated
`index`	List of integer indices of genesets, names are names of gene sets
`design`	Design matrix
`contrasts`	Contrast matrix
`doParallel`	Logical, whether `parallel::mclapply` should be used. Since at the current setting it makes a job running forever, use `TRUE` only if you are debugging the code.

Value

A `data.frame` containing CAMERA results across contrasts.

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(GeneSymbol=sprintf("Gene%d", 1:nrow(exMat)))
```

```
exPdata <- data.frame(Name=sprintf("Sample%d", 1:ncol(exMat)),
                      Group=exGroups)
exDgeList <- DGEList(exMat, genes=exFdata, samples=exPdata)
exDgeList <- edgeR::estimateDisp(exDgeList, exDesign)
cameraDGEListByContrast(exDgeList, index=1:5, design=exDesign, contrasts=exContrast)
cameraDGEListByContrast(exDgeList,
  index=list(1:5, 6:10),
  design=exDesign, contrasts=exContrast)
```

`cameraLimmaVoomResultsByContrast` *Apply the CAMERA method to a DGEList object*

Description

Apply the CAMERA method to a DGEList object

Usage

```
cameraLimmaVoomResultsByContrast(
  limmaVoomResults,
  index,
  doParallel = FALSE,
  ...
)
```

Arguments

`limmaVoomResults`	A LimmaVoomResults object, with GeneSymbol available
`index`	List of integer indices of genesets, names are names of gene sets
`doParallel`	Logical, whether `parallel::mclapply` should be used. Since at the current setting it makes a job running forever, use `TRUE` only if you are debugging the code.
`...`	Not used

Value

A `data.frame` containing CAMERA results.

A `data.frame` containing CAMERA results across contrasts.

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(GeneSymbol=sprintf("Gene%d", 1:nrow(exMat)))
exPdata <- data.frame(Name=sprintf("Sample%d", 1:ncol(exMat)),
                      Group=exGroups)
exDgeList <- DGEList(exMat, genes=exFdata, samples=exPdata)
exDgeList <- edgeR::estimateDisp(exDgeList, exDesign)
edgeObj <- EdgeObject(exDgeList, exDescon)
limmaVoomRes <- ribiosNGS::dgeWithLimmaVoom(edgeObj)
cameraLimmaVoomResultsByContrast(limmaVoomRes, index=c(1:5))
cameraLimmaVoomResultsByContrast(limmaVoomRes, index=list(GS1=1:5, GS2=6:10))
```

---

cameraTable2graph *Convert a CAMERA table into a graph*

Description

Convert a CAMERA table into a graph

Usage

```
cameraTable2graph(df, jacThr = 0.25, plot = TRUE, ...)
```

Arguments

df	Data.frame, CAMERA results
jacThr	Numeric, between 0 and 1, Jaccard Index threshold
plot	Logical, whether plotting the results
...	Passed to plot

Value

A list with two elements: graph (an igraph object) and resTbl (a data.frame with columns Namespace, GeneSet, Score).

doGse *Perform gene-set enrichment (GSE) analysis*

Description

Perform gene-set enrichment (GSE) analysis

Usage

```
doGse(edgeResult, gmtList, doParallel = FALSE)
```

Arguments

edgeResult	An object of the class EdgeResult or LimmaVoomResult
gmtList	An object of the class GmtList
doParallel	Logical, whether parallel::mclapply should be used. Since at the current setting it makes a job running forever, use TRUE only if you are debugging the code. The function performs gene-set enrichment analysis. By default,the CAMERA method is applied. In case this is not successful, for instance because of lack of biological replicates, the GAGE method (Generally Applicable Gene-set Enrichment for pathway analysis) is applied.

Value

A data.frame containing results of the gene-set enrichment analysis.

See Also

gseWithLogFCgage and gseWithCamera are wrapped by this function to perform analysis with GAGE and CAMERA, respectively. logFCgage, camera.EdgeResult, and camera.LimmaVoomResult implement the logic, and return the enrichment table.

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(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 <- ribiosNGS::dgeWithEdgeR(exObj)

exGeneSets <- BioQC::GmtList(list(
```

```
   list(name="Set1", desc="set 1", genes=c("Gene1", "Gene2", "Gene3"), namespace="default"),
   list(name="Set2", desc="set 2", genes=c("Gene18", "Gene6", "Gene4"), namespace="default")
))
exGse <- doGse(exDgeRes, exGeneSets)

## Not run:
  exMat <- matrix(rpois(120000, 10), nrow=20000, ncol=12)
  exGroups <- gl(4,3, labels=c("Group1", "Group2", "Group3", "Group4"))
  exDesign <- model.matrix(~0+exGroups)
  exContrast <- matrix(c(-1,1,0,0, 0,0,-1,1),
     ncol=2, byrow=FALSE,
     dimnames=list(c("Group1", "Group2", "Group3", "Group4"),
       c("Group2.vs.Group1", "Group4.vs.Group3")))
  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 <- ribiosNGS::dgeWithEdgeR(exObj)

  ngeneset <- 1000
  genesetSizes <- round(runif(ngeneset)*100)+1
  exGeneSets <- BioQC::GmtList(lapply(seq(1:ngeneset), function(i) {
    name <- paste0("GeneSet", i)
    desc <- paste0("GeneSet", i)
    genes <- sample(exFdata$GeneSymbol, genesetSizes[i])
    res <- list(name=name, desc=desc, genes=genes, namespace="default")
  }))
  exGse <- doGse(exDgeRes, exGeneSets)

## End(Not run)
```

`expandCameraTableGenes` *Expand genes in the CAMERA result table*

Description

Expand genes in the CAMERA result table

Usage

```
expandCameraTableGenes(tbl)
```

Arguments

`tbl`	A `data.frame`

Value

A longer `data.frame`, with each row one gene.

`factorByNumberInStr` *Make a factor vector from a character vector by the order of the parsed numbers*

Description

Make a factor vector from a character vector by the order of the parsed numbers

Usage

```
factorByNumberInStr(str, decreasing = TRUE)
```

Arguments

`str`	Strings
`decreasing`	Logical, whether decreasing or increasing order is desied, passed to `order`.

Value

A factor with levels ordered by the parsed numbers.

See Also

`orderByNumberInStr`, which returns the order of strings by numbers in them

Examples

```
factorByNumberInStr(c("D1", "D10", "D15", "D3.5"))
factorByNumberInStr(c("D1", "D10", "D15", "D3.5"), decreasing=FALSE)
```

`fdrValue` *Return FDR values*

Description

Return FDR values

Usage

```
fdrValue(object, ...)
```

Arguments

object	An object
...	Other parameters

Value

A numeric vector of FDR values.

---

filterBySize *Filter by size*

Description

Filter by size

Usage

```
filterBySize(object, min, max)
```

Arguments

object	An object
min	Integer, minimum size
max	Integer, maximum size

Value

The filtered object.

FisherResult-class *Result of Fisher's exact test*

Description

Result of Fisher's exact test

Value

An object of class `FisherResult`.

FisherResultList-class *A list of results of Fisher's exact test*

Description

A list of results of Fisher's exact test

Value

An object of class `FisherResultList`.

fishersMethod *Fisher's method to combine multiple p-values*

Description

Fisher's method to combine multiple p-values

Usage

```
fishersMethod(p, returnValiePvalues = FALSE)
```

Arguments

`p`	Numeric vector, p values to be combined
`returnValiePvalues`	Logical, whether the valid p-values used should be returned as part of the list

Value

A `FisherMethodResult` S3 object, a list of following elements

1. chisq: Chi-square statistic
2. df: Degree of freedom (which is twice the count of the valid p-values used for calculation)
3. p: p-value
4. validp (optional): valid p-values used for the calculation

The function returns the combined p-value using the sum of logs (Fisher's) method

Note

The function was adapted from metap::sumlog

Examples

```
ps <- c(0.05, 0.75)
fishersMethod(ps)
fishersMethod(ps, returnValiePvalues=TRUE)
```

---

fisherTest *Perform Fisher's exact test*

Description

Perform Fisher's exact test

Usage

```
fisherTest(genes, genesets, universe, ...)
```

Arguments

genes	Genes
genesets	Gene-sets
universe	The universe of genes
...	Other parameters

Value

A FisherResult object or a data.table of results.

---

fisherTest,character,character,character-method *Perform Fisher's exact test on a gene set*

Description

Perform Fisher's exact test on a gene set

Usage

```
## S4 method for signature 'character,character,character'
fisherTest(
  genes,
  genesets,
  universe,
  gsName,
  gsNamespace,
  makeUniqueNonNA = TRUE,
  checkUniverse = TRUE,
  useEASE = FALSE
)
```

Arguments

genes	a collection of genes of which over-representation of the gene set is tested
genesets	A vector of character strings, genes belonging to one gene set.
universe	universe of genes
gsName	gene set name, can be left missing
gsNamespace	gene set namespace name, can be left missing
makeUniqueNonNA	Logical, whether genes, geneSetGenes, and universe should be filtered to remove NA and made unique. The default is set to TRUE. When the uniqueness and absence of NA is ensured, this flag can be set to FALSE to accelerate the operation.
checkUniverse	Logical, if TRUE, then genes that are in genes but are not in universe are appended to universe
useEASE	Logical, whether to use the EASE method to report the p-value. This function performs one-sided Fisher's exact test to test the over-representation of gene set genes in the input gene list. If useEASE is TRUE, one gene is penalized (removed) within geneSetGenes that are in genes and calculating the resulting Fisher exact probability for that namespace. The theoretical basis of the EASE score lies in the concept of jackknifing a probability. See Hosack *et al.* for details.

Note

Duplicated items in genes, genesets' genes, and the universe are per default removed

References

Hosack *et al.* Hosack, Douglas A., Glynn Dennis, Brad T. Sherman, H. Clifford Lane, and Richard A. Lempicki. Identifying Biological Themes within Lists of Genes with EASE. Genome Biology 4 (2003): R70. doi:10.1186/gb2003410r70

Examples

```
myGenes <- LETTERS[1:3]
myGeneSet1 <- LETTERS[1:6]
myGeneSet2 <- LETTERS[4:7]
myUniverse <- LETTERS
fisherTest(genes=myGenes, genesets=myGeneSet1, universe=myUniverse)
fisherTest(genes=myGenes, genesets=myGeneSet2, universe=myUniverse)
fisherTest(genes=myGenes, genesets=myGeneSet1, universe=myUniverse,
           gsName="My gene set1", gsNamespace="Letters")

## note that duplicated items are removed by default
resWoRp <- fisherTest(genes=rep(myGenes,2), genesets=myGeneSet1,
                      universe=myUniverse)
resWithRp <- fisherTest(genes=rep(myGenes,2), genesets=myGeneSet1,
                      universe=rep(myUniverse,2))
```

```
identical(resWoRp, resWithRp)

resWithRpNoUnique <- fisherTest(genes=rep(myGenes,2), genesets=myGeneSet1,
           universe=rep(myUniverse,2), makeUniqueNonNA=FALSE)
identical(resWoRp, resWithRpNoUnique)
```

`fisherTest,character,GmtList,character-method`
Perform Fisher's exact test on a GmtList object

Description

Perform Fisher's exact test on a GmtList object

Usage

```
## S4 method for signature 'character,GmtList,character'
fisherTest(
  genes,
  genesets,
  universe,
  gsNamespace,
  makeUniqueNonNA = TRUE,
  checkUniverse = TRUE,
  useEASE = FALSE
)
```

Arguments

genes	character strings of gene list to be tested
genesets	An GmtList object
universe	Universe (background) gene list
gsNamespace	Character string, gene-set namespace(s)
makeUniqueNonNA	Logical, whether genes and universe should be filtered to remove NA and made unique. The default is set to TRUE. When the uniqueness and absence of NA is ensured, this flag can be set to FALSE to accelerate the operation.
checkUniverse	Logical, if TRUE, then genes that are in genes but are not in universe are appended to universe
useEASE	Logical, whether to use the EASE method to report the p-value.

Value

A `data.table` containing Fisher's exact test results of all gene-sets, in the same order as the input gene-sets, with following columns:

1. GeneSetNamespace
2. GeneSetName
3. GeneSetEffectiveSize, the count of genes in the gene-set that are found in the universe
4. HitCount, the count of genes in the `genes` input that are in the gene-set
5. Hits, a vector of character string, representing hits
6. PValue
7. FDR, PValue adjusted by the Benjamini-Hochberg method. If more than one gene-set categories are provided, the FDR correction is performed per namespace

Examples

```
gs1 <- list(name="GeneSet1", desc="desc", genes=LETTERS[1:4], namespace="A")
gs2 <- list(name="GeneSet2", desc="desc", genes=LETTERS[5:8], namespace="A")
gs3 <- list(name="GeneSet3", desc="desc", genes=LETTERS[seq(2,8,2)], namespace="A")
gs4 <- list(name="GeneSet3", desc="desc", genes=LETTERS[seq(1,7,2)], namespace="B")
gmtList <- BioQC::GmtList(list(gs1, gs2, gs3, gs4))
myInput <- LETTERS[2:6]
myUniverse <- LETTERS
myFisherRes <- fisherTest(myInput, gmtList, myUniverse)
```

`fisherTest,character,list,character-method`

Perform Fisher's exact test on a GeneSet object

Description

Perform Fisher's exact test on a GeneSet object

Usage

```
## S4 method for signature 'character,list,character'
fisherTest(
  genes,
  genesets,
  universe,
  makeUniqueNonNA = TRUE,
  checkUniverse = TRUE,
  useEASE = FALSE
)
```

Arguments

genes	a collection of genes of which over-representation of the gene set is tested
genesets	A GmtList object.
universe	universe of genes
makeUniqueNonNA	Logical, whether genes and universe should be filtered to remove NA and made unique. The default is set to TRUE. When the uniqueness and absence of NA is ensured, this flag can be set to FALSE to accelerate the operation.
checkUniverse	Logical, if TRUE, then genes that are in genes but are not in universe are appended to universe
useEASE	Logical, whether to use the EASE method to report the p-value. This function performs one-sided Fisher's exact test to test the over-representation of gene set genes in the input gene list.

Examples

```
myGenes <- LETTERS[1:3]
myS4GeneSet1 <- list(name="GeneSet1", desc="GeneSet",
    genes=LETTERS[1:6], namespace="My namespace 1")
myS4GeneSet2 <- list(name="GeneSet1", desc="GeneSet",
    genes=LETTERS[2:7], namespace="My namespace 2")
myUniverse <- LETTERS
fisherTest(myGenes, myS4GeneSet1, myUniverse)
fisherTest(myGenes, myS4GeneSet2, myUniverse)
```

---

fisherTestEdgeResult *Run Fisher's exact test on an EdgeResult object*

Description

Run Fisher's exact test on an EdgeResult object

Usage

```
fisherTestEdgeResult(
  edgeResult,
  gmtList,
  contrast,
  thr.abs.logFC = 1,
  thr.FDR = 0.05,
  minGeneSetEffectiveSize = 5,
  maxGeneSetEffectiveSize = 500,
  ...
)
```

Arguments

edgeResult	An EdgeResult object
gmtList	A GmtList or GeneSets object
contrast	Character, the contrast of interest
thr.abs.logFC	Numeric, threshold of absolute log2 fold-change to define positively and negatively regulated genes
thr.FDR	Numeric, threshold of FDR values
minGeneSetEffectiveSize	Integer, minimal number of genes of a geneset that are quantified
maxGeneSetEffectiveSize	Integer, maximal number of genes of a geneset that are quantified
...	Passed to filter to further filter the differential gene expression table (dgeTbl).

Value

A data.table containing Fisher's exact test results for positively and negatively regulated genes.

---

fisherTestResultNewHitsProp

Append NewHitsProp to the result data.table *returned by* fisherTest

---

Description

Append NewHitsProp to the result data.table returned by fisherTest

Usage

```
fisherTestResultNewHitsProp(fisherTestResults)
```

Arguments

fisherTestResults	data.table returned by fisherTest

Value

A new data.table containing all columns of the input and NewHitsProp, a new column including the proportion of new hits in the gene-set

GeMS_BASE_URL *GeMS base URL To set GeMS base URL in your environment, use 'GeMS_BASE_URL=value' in your "~/.Renviron" file*

Description

GeMS base URL To set GeMS base URL in your environment, use 'GeMS_BASE_URL=value' in your "~/.Renviron" file

Usage

```
GeMS_BASE_URL
```

Format

An object of class `character` of length 1.

GeMS_GENESETS_URL *GeMS genesets retrieval URL*

Description

GeMS genesets retrieval URL

Usage

```
GeMS_GENESETS_URL
```

Format

An object of class `character` of length 1.

GeMS_INSERT_URL *GeMS insert URL*

Description

GeMS insert URL

Usage

```
GeMS_INSERT_URL
```

Format

An object of class `character` of length 1.

GeMS_REMOVE_URL *GeMS remove URL*

Description

GeMS remove URL

Usage

```
GeMS_REMOVE_URL
```

Format

An object of class `character` of length 1.

GeMS_TEST_GENESETS_URL *GeMS geneset retrieval URL for testing*

Description

GeMS geneset retrieval URL for testing

Usage

```
GeMS_TEST_GENESETS_URL
```

Format

An object of class `character` of length 1.

GeMS_TEST_URL *GeMS URL for testing*

Description

GeMS URL for testing

Usage

```
GeMS_TEST_URL
```

Format

An object of class `character` of length 1.

geneSetPerm *Test gene set enrichment by permutating gene labels of statistics*

Description

Test gene set enrichment by permutating gene labels of statistics

Usage

```
geneSetPerm(stats, indList, Nsim = 9999)
```

Arguments

stats	Statistics
indList	a list of integers, indicating indices of genes of gene sets (index starts from 1, following R's convention)
Nsim	number of simulations

Value

A data frame containg mean statistic, gene set size, and p-values

See Also

geneSetTest, a R implementation in the limma package

Examples

```
set.seed(1887)
stats <- rnorm(1000)
gsList <- list(gs1=c(3,4,5), gs2=c(7,8,9))
geneSetPerm(stats, gsList, Nsim=99)
gsList2 <- list(gs1=c(3,4,5), gs2=c(7,8,9), gs3=integer())
geneSetPerm(stats, gsList2, Nsim=99)
gsList3 <- sample(1:1000, 200)
geneSetPerm(stats, gsList3, Nsim=99)
```

GeneSetResult-class *A generic, virtual S4 class for gene-set analysis result*

Description

A generic, virtual S4 class for gene-set analysis result

Value

An object of class GeneSetResult (virtual).

getFDRCol — *Get the name of the column which store false-discovery rates (adjusted P-values) from topTables*

Description

Get the name of the column which store false-discovery rates (adjusted P-values) from topTables

Usage

```
getFDRCol(colnames)
```

Arguments

colnames	A character string vector of column names

Value

The column name of the FDRs, NA if not found.

Examples

```
getFDRCol(c("Feature", "logFC", "PValue", "FDR"))
getFDRCol(c("Feature", "logFC", "P.Value", "FDR"))
getFDRCol(c("Feature", "logFC", "p.Value", "adjPvalue"))
getFDRCol(c("Feature", "logFC", "PValue", "adj.PValue"))
```

getJsonResponse — *Send a list as JSON query to an URL and fetch the response*

Description

Send a list as JSON query to an URL and fetch the response

Usage

```
getJsonResponse(url, body)
```

Arguments

url	The destination URL
body	A list to be sent to the URL, which will be encoded in the JSON format internally

Value

The response from the webserver

Examples

```
## Not run:
   ## getJsonResponse(GeMS_GENESETS_URL, list(user=ribiosUtils::whoami()))

## End(Not run)
```

`getPvalCol` *Get the name of the column which store unadjusted P-values from topTables*

Description

Get the name of the column which store unadjusted P-values from topTables

Usage

```
getPvalCol(colnames)
```

Arguments

`colnames`	A character string vector of column names

Value

The column name of the unadjusted p-values, `NA` if not found.

Examples

```
getPvalCol(c("Feature", "logFC", "PValue", "FDR"))
getPvalCol(c("Feature", "logFC", "P.Value", "FDR"))
getPvalCol(c("Feature", "logFC", "p.Value", "adjPvalue"))
getPvalCol(c("Feature", "logFC", "pval", "adjPvalue"))
```

`getSetsWithNamesFromGeMS` *Get one or more gene-sets with their names*

Description

Get one or more gene-sets with their names

Usage

```
getSetsWithNamesFromGeMS(setNames = NULL)
```

Arguments

setNames	Character strings

Value

A GmtList object

See Also

getSetWithNameFromGeMS

Examples

```
## Not run:
getSetsWithNamesFromGeMS(c("Plasma_sc", "Bcell_l_Danaher17"))

## End(Not run)
```

---

getSetsWithPropertyFromGeMS
Get gene-sets for application

Description

Get gene-sets for application

Usage

```
getSetsWithPropertyFromGeMS(property = "meta.application", value = "")
```

Arguments

property	Character string, property to query
value	Character string, property value

Value

A GmtList object

Examples

```
## Not run:
getSetsWithPropertyFromGeMS("meta.application", "rtbeda_CIT")

## End(Not run)
```

getSetWithNameFromGeMS *Get one gene-set with its name*

Description

Get one gene-set with its name

Usage

```
getSetWithNameFromGeMS(setName)
```

Arguments

setName	Character string

Value

A list of two elements

1. name
2. genes

See Also

getSetsWithNamesFromGeMS

Examples

```
## Not run:
getSetWithNameFromGeMS("Plasma_sc")

## End(Not run)
```

getUserSetsFromGeMS *Get gene sets of a user from GeMS*

Description

Get gene sets of a user from GeMS

Usage

```
getUserSetsFromGeMS(user = ribiosUtils::whoami())
```

Arguments

user	User name

Value

A data.frame including following columns:

1. setName
2. desc
3. domain
4. source
5. subtype

Examples

```
## Not run:
#### my gene-sets
## getUserSetsFromGeMS()
#### from another user
## getUserSetsFromGeMS("kanga6")

## End(Not run)
```

---

gseaCoreEnrichGenes *Return GSEA core enrichment genes (also known as leading-edge genes)*

Description

Return GSEA core enrichment genes (also known as leading-edge genes)

Usage

```
gseaCoreEnrichGenes(object)

## S4 method for signature 'AnnoBroadGseaResItem'
gseaCoreEnrichGenes(object)

## S4 method for signature 'AnnoBroadGseaRes'
gseaCoreEnrichGenes(object)
```

Arguments

object	An object

Value

A character vector of core enrichment genes.

Methods (by class)

- gseaCoreEnrichGenes(AnnoBroadGseaResItem): Return core enriched genes (also known as leading-edge genes) in an AnnoBroadGseaResItem object as a character string vector.
- gseaCoreEnrichGenes(AnnoBroadGseaRes): Return core enriched genes (also known as leading-edge genes) in an AnnoBroadGseaRes object as a list of character string vectors.

---

gseaCoreEnrichThr *Return GSEA core enrichment score threshold*

Description

Return GSEA core enrichment score threshold

Usage

```
gseaCoreEnrichThr(object)

## S4 method for signature 'BroadGseaResItem'
gseaCoreEnrichThr(object)

## S4 method for signature 'AnnoBroadGseaRes'
gseaCoreEnrichThr(object)
```

Arguments

object	An object

Value

A numeric value.

Methods (by class)

- gseaCoreEnrichThr(BroadGseaResItem): Get the threshold value of GSEA core enrichment from a BroadGseaResItem object
- gseaCoreEnrichThr(AnnoBroadGseaRes): Get the threshold value of GSEA core enrichment from an AnnoBroadGseaRes object

gseaES *Return GSEA enrichment scores*

Description

Return GSEA enrichment scores

Usage

```
gseaES(object)

## S4 method for signature 'BroadGseaResItem'
gseaES(object)

## S4 method for signature 'AnnoBroadGseaRes'
gseaES(object)

## S4 method for signature 'AnnoBroadGseaResList'
gseaES(object)
```

Arguments

object	An object

Value

A numeric vector of enrichment scores.

Methods (by class)

- gseaES(BroadGseaResItem): Get GSEA enrichment score from a BroadGseaResItem object
- gseaES(AnnoBroadGseaRes): Get GSEA enrichment score from an AnnoBroadGseaRes object
- gseaES(AnnoBroadGseaResList): Get GSEA enrichment score from an AnnoBroadGseaResList object

gseaESprofile *Return GSEA enrichment score profile*

Description

Return GSEA enrichment score profile

Usage

```
gseaESprofile(object)

## S4 method for signature 'BroadGseaResItem'
gseaESprofile(object)
```

Arguments

object	An object

Value

A numeric vector of enrichment score profiles.

Methods (by class)

- `gseaESprofile(BroadGseaResItem)`: Get GSEA enrichment profile from a BroadGseaResItem object

---

gseaFDR *Return GSEA FDR*

Description

Return GSEA FDR

Usage

```
gseaFDR(object)

## S4 method for signature 'BroadGseaResItem'
gseaFDR(object)

## S4 method for signature 'AnnoBroadGseaRes'
gseaFDR(object)

## S4 method for signature 'AnnoBroadGseaResList'
gseaFDR(object)
```

Arguments

object	An object

Value

A numeric vector of FDR values.

Methods (by class)

- `gseaFDR(BroadGseaResItem)`: Get GSEA FDR values from a BroadGseaResItem object
- `gseaFDR(AnnoBroadGseaRes)`: Get GSEA FDR values from an AnnoBroadGseaRes object
- `gseaFDR(AnnoBroadGseaResList)`: Get GSEA FDR values from an AnnoBroadGseaResList object

---

gseaFingerprint *Extract pathway fingerprints from GSEA results*

Description

`gseaFingerprint` extracts pathway fingerprints from the result of one GSEA result. `gseaFingerprintMatrix` extracts multiple signatures and organizes into the form of rectangular matrix.

Usage

```
gseaFingerprint(
  gseaDir,
  value = c("q", "es", "nes"),
  threshold = 1e-04,
  sortByName = TRUE
)

gseaFingerprintMatrix(gseaDirs, value = c("q", "es", "nes"), ...)
```

Arguments

`gseaDir`	Character, a GSEA output directory. Notice the directory must be accessible by the R session. A common mistake is to use a relative path which cannot be found.
`value`	Character, the statistic to extract, currently supporting `q`, `es` and `nes`
`threshold`	Numeric, minimum threshold of q-value, passed to `gseaQvalue`
`sortByName`	Logical, whether signatures should be sorted by name
`gseaDirs`	Character vector, GSEA output directories
`...`	Parameters passed to `gseaFingerprint` by `gseaFingerprintMatrix`

Details

`gseaFingerprint` extracts pathway signature from one GSEA output directory. While `gseaFingerprintMatrix` simultaneously extracts from more than one GSEA output directories, and organizes pathway signatures in a rectangular matrix form.

`gseaFingerprintMatrix` takes care of signature mapping between different GSEA result sets.

Value

gseaFingerprint returns a data.frame with two columns name and value, recording gene signature (pathway) names and the statistic chosen by the user.

gseaFingerprintMatrix returns a matrix, with the union set of gene signatures from all GSEA output result sets as rows, and GSEA result names as columns.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

See gseaQvalue and gseaES for how to choose the statistic to produce pathway signatures.

Examples

```
gseaDirZip <- system.file(package="ribiosGSEA","extdata/gseaDirs.zip")
tmpDir <- tempdir()
utils::unzip(gseaDirZip, exdir=tmpDir)
gseaDir <- file.path(tmpDir, "gseaDirs")
gseaDirs <- dir(gseaDir, full.names=TRUE)
gseaFp <- gseaFingerprint(gseaDirs[1], value="q")
gseaFps <- gseaFingerprintMatrix(gseaDirs, value="q")
```

---

gseaFWER *Return GSEA FWER values*

Description

Return GSEA FWER values

Usage

```
gseaFWER(object)

## S4 method for signature 'BroadGseaResItem'
gseaFWER(object)

## S4 method for signature 'AnnoBroadGseaRes'
gseaFWER(object)

## S4 method for signature 'AnnoBroadGseaResList'
gseaFWER(object)
```

Arguments

object	An object

Value

A numeric vector of FWER values.

Methods (by class)

- `gseaFWER(BroadGseaResItem)`: Get GSEA FWER values from a BroadGseaResItem object
- `gseaFWER(AnnoBroadGseaRes)`: Get GSEA FWER values from an AnnoBroadGseaRes object
- `gseaFWER(AnnoBroadGseaResList)`: Get GSEA FWER values from an AnnoBroadGseaResList object

---

gseaNES *Return GSEA normalized enrichment scores*

Description

Return GSEA normalized enrichment scores

Usage

```
gseaNES(object)

## S4 method for signature 'BroadGseaResItem'
gseaNES(object)

## S4 method for signature 'AnnoBroadGseaRes'
gseaNES(object)

## S4 method for signature 'AnnoBroadGseaResList'
gseaNES(object)
```

Arguments

`object`	An object

Value

A numeric vector of normalized enrichment scores.

Methods (by class)

- `gseaNES(BroadGseaResItem)`: Get GSEA normalized enrichment score from a BroadGseaResItem object
- `gseaNES(AnnoBroadGseaRes)`: Get GSEA normalized enrichment score from an AnnoBroadGseaRes object
- `gseaNES(AnnoBroadGseaResList)`: Get GSEA normalized enrichment score from an AnnoBroadGseaResList object

gseaNP *Return GSEA number of permutation*

Description

Return GSEA number of permutation

Usage

```
gseaNP(object)

## S4 method for signature 'BroadGseaResItem'
gseaNP(object)

## S4 method for signature 'AnnoBroadGseaRes'
gseaNP(object)

## S4 method for signature 'AnnoBroadGseaResList'
gseaNP(object)
```

Arguments

object	An object

Value

A numeric vector.

Methods (by class)

- gseaNP(BroadGseaResItem): Get GSEA number of permutations from a BroadGseaResItem object
- gseaNP(AnnoBroadGseaRes): Get GSEA number of permutations from an AnnoBroadGseaRes object
- gseaNP(AnnoBroadGseaResList): Get GSEA number of permutations from an AnnoBroadGseaResList object

gseaResQvalue *Read GSEA statistic for pathway fingerprinting*

Description

Read GSEA statistics (log-transformed q-value [q], Enrichment Score [ES], or normalized Enrichement Score [NES]) to profile pathway activitities.

Usage

```
gseaResQvalue(file, threshold = 1e-04, log = FALSE, posLog = FALSE)

gseaResES(file, normalized = FALSE)
```

Arguments

file	GSEA output tab-delimited file, usually with the file name 'gsea_report_for.*_pos_.*.xls' or 'gsea_report_for.*_neg_.*.xls'. Located in GSEA output directory.
threshold	Valid for q value: what is the minimum threshold of q-value (FDR)? It can be set to the number of permutation tests divided by 1. By default 1/10000
log	Valid for q value: whether the FDR q value should be transformed by base-10 (log10) logarithm. By default FALSE
posLog	Valid for q value: whether the logged FDR q value should be negated to get positive value.This is useful when the sign of q is used to distinguish between positive and negative enriched pathways. By default FALSE.
normalized	Valid for enrichment score: if set to TRUE, normalized enrichment score (nes) will be returned instead of (es). By default set to FALSE

Details

In many cases we want to extract pathway signatures from a set of experiments. Both gseaResQvalue and gseaES can read GSEA output files and extract desired statistic: q-value, ES or NES.

See the GSEA document for definitions of the three values. For comparing a few conditions to another, we recommend using *q-value*. For large-scale comparisons between pathways (or other gene signatures), we have found *ES* very useful. It is adviced to choose proper statistic to extract pathway signatures only when you are sure of the aim. Using any statistic without good reasoning may as always lead to wrong intepretations of the data.

These functions are usually not directly called by end-users. See gseaFingerprint and link{gseaFingerprintMatrix} instead.

Value

A data.frame with two columns: name and value. The column name contains gene signatures (e.g. pathways), and value contains the statistic.

Functions

- `gseaResQvalue()`: The function to extract the Q-value
  Extract Q-values from GSEA result file

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>, with input from Martin Ebeling, Laura Badi and Isabelle Wells.

References

GSEA documentation http://www.broadinstitute.org/gsea/doc/GSEAUserGuideFrame.html

See Also

End-users will probably find `gseaFingerprint` and `link{gseaFingerprintMatrix}` more useful, since they operate on the level of GSEA result directories, instead of single output tab-delimited files.

Examples

```
gseaDirZip <- system.file(package="ribiosGSEA","extdata/gseaDirs.zip")
tmpDir <- tempdir()
utils::unzip(gseaDirZip, exdir=tmpDir)
gseaDir <- file.path(tmpDir, "gseaDirs")
gseaFile <- file.path(gseaDir,
   "VitaminA_24h_High",
   "gsea_report_for_na_neg_1336489010730.xls")

gseaQ <- gseaResQvalue(gseaFile)
gseaLogQ <- gseaResQvalue(gseaFile, log=TRUE)
gseaQscore <- gseaResQvalue(gseaFile, log=TRUE, posLog=TRUE)

gseaEs <- gseaResES(gseaFile)
gseaNes <- gseaResES(gseaFile, normalized=TRUE)
```

---

gseaScore *Extract scores from GSEA results*

Description

One way to score GSEA results is to multiple the absolute value of log10 transformed p-values (nominal p-value, FDR, or FWER) with the sign of the enrichment scores. This score is intuitive since it combines statistical significance and the sign of regulation.

Usage

```
gseaScore(x, type = c("fdr", "p", "fwer"))

gseaScores(..., names = NULL, type = c("fdr", "p", "fwer"))
```

Arguments

x	An AnnoBroadGseaRes object
type	Character string, the type of p-value used to calculate the score.
...	Objects of AnnoBroadGseaRes to be compared
names	Character strings, names given to the result score sets. See examples below.

Details

gseaScores takes care of the situation where some gene sets are missing in one or more conditions.

Value

gseaScore returns a double vector of scores with gene set names.

gseaScores returns a data frame of scores, with gene set names as row names.

Functions

- gseaScores(): gseaScore applied to multiple objects

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

See Also

gseaNP, gseaFDR, gseaFWER to get p-values.

parseGSEAdir.

---

gsEffectiveSize *Return the effective size of gene-set*

Description

Return the effective size of gene-set

Usage

```
gsEffectiveSize(object, ...)

## S4 method for signature 'FisherResult'
gsEffectiveSize(object)

## S4 method for signature 'FisherResultList'
gsEffectiveSize(object)
```

Arguments

object	An object
...	Other parameters

Value

An integer vector of effective sizes.

Methods (by class)

- gsEffectiveSize(FisherResult): Effective sizes of gene-set, returning an integer.
- gsEffectiveSize(FisherResultList): Effective sizes of Gene-sets, returning an integer vector.

---

gsFisherTestCore *The core algorithm to perform Fisher's exact test on a gene set*

Description

The core algorithm to perform Fisher's exact test on a gene set

Usage

```
gsFisherTestCore(
  genes,
  geneSetGenes,
  universe,
  makeUniqueNonNA = TRUE,
  checkUniverse = TRUE,
  useEASE = FALSE
)
```

Arguments

genes	Character vector, a collection of genes of which over-representation of the gene set is tested
geneSetGenes	Character vector, genes belonging to a gene set
universe	Character vector, universe of genes
makeUniqueNonNA	Logical, whether genes, geneSetGenes, and universe should be filtered to remove NA and made unique. The default is set to TRUE. When the uniqueness and absence of NA is ensured, this flag can be set to FALSE to accelerate the operation.
checkUniverse	Logical, if TRUE, then genes that are in genes but are not in universe are appended to universe
useEASE	Logical, whether to use the EASE method to report the p-value. This function performs one-sided Fisher's exact test to test the over-representation of the genes given as geneSetGenes in the input genes list. If useEASE is TRUE, one gene is penalized (removed) within geneSetGenes that are in genes and calculating the resulting Fisher exact probability for that namespace. The theoretical basis of the EASE score lies in the concept of jackknifing a probability. See Hosack *et al.* for details.

Value

A list of three elements

1. p The p-value of one-sided (over-representation of the Fisher's test)
2. gsEffectiveSize Gene-set's effective size, namely number of genes that are in the universe
3. hits Character vector, genes that are found in the gene sets

References

Hosack *et al.* Hosack, Douglas A., Glynn Dennis, Brad T. Sherman, H. Clifford Lane, and Richard A. Lempicki. Identifying Biological Themes within Lists of Genes with EASE. Genome Biology 4 (2003): R70. doi:10.1186/gb2003410r70

Examples

```
myGenes <- LETTERS[1:3]
myGeneSet1 <- LETTERS[1:6]
myGeneSet2 <- LETTERS[4:7]
myUniverse <- LETTERS
gsFisherTestCore(myGenes, myGeneSet1, myUniverse)
gsFisherTestCore(myGenes, myGeneSet2, myUniverse)

## use EASE for conservative estimating
gsFisherTestCore(myGenes, myGeneSet1, myUniverse, useEASE=FALSE)
gsFisherTestCore(myGenes, myGeneSet1, myUniverse, useEASE=TRUE)

## checkUniverse will make sure that \code{univese} contains all element in \code{genes}
```

```
gsFisherTestCore(c("OutOfUniverse", myGenes), myGeneSet1, myUniverse, checkUniverse=FALSE)
gsFisherTestCore(c("OutOfUniverse", myGenes), myGeneSet1, myUniverse, checkUniverse=TRUE)
```

`gsGeneCount` *Return gene-set gene count*

Description

Return gene-set gene count

Usage

```
gsGeneCount(object, ...)
```

Arguments

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

Value

An integer vector of gene counts.

`gsGeneIndices` *Return gene-set gene indices*

Description

Return gene-set gene indices

Usage

```
gsGeneIndices(object)

## S4 method for signature 'BroadGseaResItem'
gsGeneIndices(object)
```

Arguments

`object`	An object

Value

An integer vector of gene indices.

Methods (by class)

- `gsGeneIndices(BroadGseaResItem)`: Get gene-set gene indices from a BroadGseaResItem object, returning a vector of integers.

gsGenes *Return gene-set genes*

Description

Return gene-set genes

Usage

```
gsGenes(object, ...)

## S4 method for signature 'AnnoBroadGseaResItem'
gsGenes(object)

## S4 method for signature 'AnnoBroadGseaRes'
gsGenes(object)

## S4 method for signature 'GmtList'
gsGenes(object)
```

Arguments

object	An object
...	Other parameters

Value

A character vector or list of character vectors of gene-set genes.

Methods (by class)

- gsGenes(AnnoBroadGseaResItem): Get gene-set genes from a BroadGseaResItem object, returning a character string vector.
- gsGenes(AnnoBroadGseaRes): Get gene-set genes from an AnnoBroadGseaRes object, returning a list of character string vectors.
- gsGenes(GmtList): Get gene-set genes from a GmtList object, returning a list of character string vector. It uses the implementation in BioQC.

gsGenes<- *gsGenes-set*

Description

Set gene-set genes

Usage

```
gsGenes(object) <- value

## S4 replacement method for signature 'AnnoBroadGseaResItem,character'
gsGenes(object) <- value
```

Arguments

object	An object
value	Value

Value

The modified object.

Functions

- gsGenes(object = AnnoBroadGseaResItem) <- value: Assign gene-set genes to AnnoBroadGseaResItem

gsGeneValues *Return gene-set gene values*

Description

Return gene-set gene values

Usage

```
gsGeneValues(object)

## S4 method for signature 'AnnoBroadGseaResItem'
gsGeneValues(object)

## S4 method for signature 'AnnoBroadGseaRes'
gsGeneValues(object)
```

Arguments

object	An object

Value

A numeric vector or list of numeric vectors of gene values.

Methods (by class)

- gsGeneValues(AnnoBroadGseaResItem): Return values associated with the genes in a gene-set in an AnnoBroadGseaResItem object in a numeric vector.
- gsGeneValues(AnnoBroadGseaRes): Return values associated with the genes in a gene-set in an AnnoBroadGseaRes object in a list of numeric vectors.

---

gsGeneValues<- *gsGeneValues-set*

Description

Set gene-set gene statistics (values)

Usage

```
gsGeneValues(object) <- value

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

Arguments

object	An object
value	Value

Value

The modified object.

Functions

- gsGeneValues(object = AnnoBroadGseaResItem) <- value: Assign values associated with gene-set genes to an annoBraoadGseaResItem object

`gsListFisherTestCore` *Core algorithm to perform Fisher's exact test on a list of gene set*

Description

Core algorithm to perform Fisher's exact test on a list of gene set

Usage

```
gsListFisherTestCore(
  genes,
  geneSetGenesList,
  universe,
  makeUniqueNonNA = TRUE,
  checkUniverse = TRUE,
  useEASE = FALSE
)
```

Arguments

`genes`	Character vector, a collection of genes of which over-representation of the gene set is tested
`geneSetGenesList`	A list of character vector, genes belonging to each gene set
`universe`	Character vector, universe of genes
`makeUniqueNonNA`	Logical, whether genes, geneSetGenes, and universe should be filtered to remove NA and made unique. The default is set to `TRUE`. When the uniqueness and absence of NA is ensured, this flag can be set to `FALSE` to accelerate the operation.
`checkUniverse`	Logical, if `TRUE`, then genes that are in `genes` but are not in `universe` are appended to `universe`
`useEASE`	Logical, whether to use the EASE method to report the p-value. This function performs one-sided Fisher's exact test to test the over-representation of the genes given as `geneSetGenes` in the input `genes` list. If `useEASE` is `TRUE`, one gene is penalized (removed) within `geneSetGenes` that are in `genes` and calculating the resulting Fisher exact probability for that namespace. The theoretical basis of the EASE score lies in the concept of jackknifing a probability. See Hosack *et al.* for details.

Value

A list of lists, of the same length as the input geneSetGenesList, each list consisting of three elements

1. p The p-value of one-sided (over-representation of the Fisher's test)

2. gsEffectiveSize Gene-set's effective size, namely number of genes that are in the universe
3. hits Character vector, genes that are found in the gene sets

References

Hosack *et al.* Hosack, Douglas A., Glynn Dennis, Brad T. Sherman, H. Clifford Lane, and Richard A. Lempicki. Identifying Biological Themes within Lists of Genes with EASE. Genome Biology 4 (2003): R70. doi:10.1186/gb2003410r70

See Also

gsFisherTestCore

Examples

```
myGenes <- LETTERS[1:3]
myGeneSet1 <- LETTERS[1:6]
myGeneSet2 <- LETTERS[4:7]
myUniverse <- LETTERS
gsListFisherTestCore(myGenes, list(myGeneSet1, myGeneSet2), myUniverse)
```

---

gsName *Return gene-set name*

Description

Return gene-set name

Usage

```
gsName(object, ...)

## S4 method for signature 'BroadGseaResItem'
gsName(object)

## S4 method for signature 'AnnoBroadGseaRes'
gsName(object)

## S4 method for signature 'FisherResult'
gsName(object)

## S4 method for signature 'FisherResultList'
gsName(object, ...)

## S4 method for signature 'GmtList'
gsName(object)

## S4 method for signature 'FisherResultList'
gsName(object, ...)
```

Arguments

object	An object
...	Other parameters

Value

A character vector of gene-set names.

Methods (by class)

- gsName(BroadGseaResItem): Get gene-set name from a BroadGseaResItem object
- gsName(AnnoBroadGseaRes): Get gene-set name from an AnnoBroadGseaRes object
- gsName(FisherResult): Get gene-set name from a FisherResult object
- gsName(FisherResultList): Get gene-set name from a FisherResultList object
- gsName(GmtList): Get gene-set name from a GmtList object
- gsName(FisherResultList): Get gene-set name from a FisherResultList object

---

gsNamespace *Return gene-set namespace*

Description

Return gene-set namespace

Usage

```
gsNamespace(object, ...)

## S4 method for signature 'GmtList'
gsNamespace(object)

## S4 method for signature 'FisherResult'
gsNamespace(object)

## S4 method for signature 'FisherResultList'
gsNamespace(object)
```

Arguments

object	An object
...	Other parameters

Value

A character vector of gene-set namespaces.

Methods (by class)

- `gsNamespace(GmtList)`: Return gene-set namespace from a GmtList object
- `gsNamespace(FisherResult)`: Return gene-set namespace from a FisherResult object
- `gsNamespace(FisherResultList)`: Return gene-set namespace from a FisherResultList object.

`gsSize` *Return the size (unique length) of gene-sets*

Description

Return the size (unique length) of gene-sets

Usage

```
gsSize(gmtList)
```

Arguments

`gmtList`	a `GmtList` object

Value

An integer vector

`hits` *Return hits*

Description

Return hits

Usage

```
hits(object, ...)

## S4 method for signature 'FisherResult'
hits(object)

## S4 method for signature 'FisherResultList'
hits(object, geneset)
```

Arguments

object	An object
...	Other parameters
geneset	Character string, gene-set name

Value

A character vector or list of hit genes.

Methods (by class)

- hits(FisherResult): Return hits from a FisherResult object
- hits(FisherResultList): Return hits from a FisherResultList object, returning a list if geneset is missing, or gene-set genes if geneset is present.

---

insertGmtListToGeMS *Insert a GmtList object to GeMS*

Description

Insert a GmtList object to GeMS

Usage

```
insertGmtListToGeMS(
  gmtList,
  geneFormat = 0,
  source = "PubMed",
  taxID = 9606,
  user = ribiosUtils::whoami(),
  subtype = "",
  domain = ""
)
```

Arguments

gmtList	A GmtList object defined in the BioQC package
geneFormat	Integer index of gene format. 0 stands for official human gene symbol
source	Character, source of the gene set
taxID	Integer, NCBI taxonomy ID of the species.
user	The user name
subtype	Subtype of the geneset
domain	Domain of the geneset

Value

Response code or error message returned by the GeMS API. A value of `200` indicates a successful insertion.

See Also

`removeFromGeMS`

Examples

```
## Not run:
  testList <- list(list(name="GS_A", desc=NULL, genes=c("MAPK14", "JAK1", "EGFR")),
    list(name="GS_B", desc="gene set B", genes=c("ABCA1", "DDR1", "DDR2")),
    list(name="GS_C", desc="gene set C", genes=NULL))
  testGmt <- BioQC::GmtList(testList)
  ## insertGmtListToGeMS(testGmt, geneFormat=0, source="Test")
  ## removeFromGeMS(setName=c("GS_A", "GS_B", "GS_C"), source="Test")

## End(Not run)
```

`insertGmtListToGeMSBody` *Construct message body to insert into GeMS*

Description

Construct message body to insert into GeMS

Usage

```
insertGmtListToGeMSBody(
  gmtList,
  geneFormat = 0,
  source = "PubMed",
  taxID = 9606,
  user = ribiosUtils::whoami(),
  subtype = "",
  domain = ""
)
```

Arguments

`gmtList`	A `GmtList` object defined in the `BioQC` package
`geneFormat`	Integer index of gene format. 0 stands for official human gene symbol
`source`	Character, source of the gene set
`taxID`	Integer, NCBI taxonomy ID of the species.

user	The user name
subtype	Subtype of the geneset
domain	Domain of the geneset

Value

A list with three items: headers, parsed, and params

Examples

```
testList <- list(list(name="GS_A", desc=NULL, genes=c("MAPK14", "JAK1", "EGFR")),
  list(name="GS_B", desc="gene set B", genes=c("ABCA1", "DDR1", "DDR2")),
  list(name="GS_C", desc="gene set C", genes=NULL))
testGmt <- BioQC::GmtList(testList)
insertGmtListToGeMSBody(testGmt, geneFormat=0, source="Test")
```

---

isGeMSReachable *Test whether GeMS is reachable*

Description

Test whether GeMS is reachable

Usage

```
isGeMSReachable()
```

Value

Logical value

Examples

```
## Not run:
  ## isGeMSReachble()

## End(Not run)
```

isGseaCoreEnrich — *Return a vector of logical values, indicating whether genes belong to core enrichment or not*

Description

Return a vector of logical values, indicating whether genes belong to core enrichment or not

Usage

```
isGseaCoreEnrich(object)
```

Arguments

object	An AnnoBroadGseaResItem object

Value

A logical vector

isSigGeneSet — *Return a logical vector indicating whether a gene-set is significantly enriched or not, given the FDR threshold*

Description

Return a logical vector indicating whether a gene-set is significantly enriched or not, given the FDR threshold

Usage

```
isSigGeneSet(object, fdr = 0.05)
```

Arguments

object	A FisherResultList object
fdr	Numeric, FDR value threshold

Value

A logical vector

kendallW *S3 generic for kendallW*

Description

S3 generic for kendallW

Usage

```
kendallW(object, ...)
```

Arguments

object	An object
...	Other parameters

Value

A matrix with an info attribute, see kendallWmat.

kendallW.eSet *Compute Kendall's W for an eSet object*

Description

Compute Kendall's W for an eSet object

Usage

```
## S3 method for class 'eSet'
kendallW(
  object,
  row.factor,
  summary = c("none", "mean", "median", "max.mean.sig", "max.var.sig"),
  na.rm = TRUE,
  alpha = 0.01,
  ...
)
```

Arguments

object	An eSet object
row.factor	A factor indicating groups of rows. In expression analysis, for instance, this can be GeneIDs indicating which probesets in rows belong to the same gene.
summary	Summary type, passed to kendallWmat
na.rm	Logical, whether NA values should be removed
alpha	Numeric, passed to kendallWmat
...	Not used

Value

An ExpressionSet object with consolidated features.

See Also

kendallWmat

---

kendallW.matrix *Compute Kendall's W for a matrix*

Description

Compute Kendall's W for a matrix

Usage

```
## S3 method for class 'matrix'
kendallW(
  object,
  row.factor,
  summary = c("none", "mean", "median", "max.mean.sig", "max.var.sig"),
  na.rm = TRUE,
  alpha = 0.01,
  ...
)
```

Arguments

object	A numeric matrix
row.factor	A factor indicating groups of rows. In expression analysis, for instance, this can be GeneIDs indicating which probesets in rows belong to the same gene.
summary	Summary type, passed to kendallWmat
na.rm	Logical, whether NA values should be removed
alpha	Numeric, passed to kendallWmat
...	Not used

Value

A matrix with an `info` attribute, see `kendallWmat`.

See Also

`kendallWmat`

`kendallWinfo` *S3 generic for kendallW information*

Description

S3 generic for kendallW information

Usage

```
kendallWinfo(object)

## S3 method for class 'matrix'
kendallWinfo(object)
```

Arguments

`object`	An object

Value

A `data.frame` containing grouping information.

Methods (by class)

- `kendallWinfo(matrix)`: Extract kendallW information from a matrix

`kendallWinfo<-.matrix` *S3 method to assign kendallW information to a matrix*

Description

S3 method to assign kendallW information to a matrix

Usage

```
## S3 replacement method for class 'matrix'
kendallWinfo(object) <- value
```

Arguments

object	matrix
value	assigned value

Value

The matrix containing grouping information

---

kendallWmat *Use Kendall's W and graph theory to assign independent measurements into sub-groups by correlation*

Description

Kendall's W, also known as Kendall's coefficient of concordance, is a non-parametric statistic developed to assess agreement among raters used in psychological or similar experimental settings.

Usage

```
kendallWmat(
  mat,
  row.factor,
  summary = c("none", "mean", "median", "max.mean.sig", "max.var.sig"),
  na.rm = TRUE,
  alpha = 0.01
)
```

Arguments

mat	A numeric matrix. It must contain at least 2 rows and 2 columns.
row.factor	A factor indicating groups of rows. In expression analysis, for instance, this can be GeneIDs indicating which probesets in rows belong to the same gene.
summary	Character, action to take once the sub-groups have been determined. 'none' indicates no action should be taken, the original data is returned with the information of sub-grouping. The option 'mean' (or 'median') will take mean/median of features in each sub-group as result. On contrast, max.mean.sig or max.var.sig picks the feature of the largest mean signal or the largest variance in each sub-group as the representative. See details below.
na.rm	Logical, should those features whose row.factor are NA be left out? If set to TRUE (which is default), these unannotated features will be discarded from the results.
alpha	Nunmeric value, the significance level of the Kendall's W statistic. The larger the value, the more abbreviations from strong associations are allowed in sub-groups. Default is 0.01.

Details

In computational biology, the concept of associating features with similar patterns while keeping outliers can be useful in many cases. See the Details section for examples.

This function implements the Kendall's W recursively with graph theory. It split grouped measurements into strongly associated sub-groups. See the Details section.

We take a microarray experiment as an example to demonstrate how the function works. In microarrays, a gene is often represented by more than one probeset, and it is not rare that they do not all resemble the same expression pattern. Usually a `one gene-one value` relation is desired. Common practices including choosing the probeset with the highest average signal or the highest variance, as well as taking the mean/median value of all probesets mapped to one gene as the representative value.

Kendall's W takes a very different approach. First it tries to judge whether multiple probesets of one gene are concordant. The concordance is determined by a non-parametric statistic closely related to Spearman correlation coefficient as well as Friedman's test. If all probesets are concordant, it means that their expression patterns are closely associated with each other. Any one of them, or the mean value, can be then used to represent the expression level of the gene.

In cases where there is little concordance among probesets, we can take use of graph theory to iteratively search for sub-groups of probesets resemble each other's expression patterns. In the extreme case, each probeset can be different from the rest, and in this case the number of sub-groups will be equal to the number of probesets mapped to the gene. Such cases can appear, for instance, when each probeset was designed to target a different region of a transcript with splice variants. By using Kendall's W statistic with graph theory, the `kendallWmat` function can detect sub-groups with strongly correlated expression patterns, while keeping outliers on their own, therefore providing help for both conventional expression analysis and post-hoc analysis with the help of sequence analysis. See reference for examples on this application.

We believe this approach is only useful for microarray, but can be also interesting for other applications like next-generation sequencing (NGS) or pathway/network analysis. For instance, in NGS experiments, this method can help to determine which splice variants of a transcript have similar expression patterns, and how different are other variants. In pathway analysis, when rows indicate gene expression values and `row.factor` indicate pathway membership, the result reveals which sub-networks are regulated associatively.

Value

Currently a matrix with one attribute slot named `info`.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

References

The concept of Kendall's W was introduced in the seminal paper *The problem of m rankings* by M.G. Kendall and B.B. Smith (The Annals of Mathematical Statistics, 1939). Schneider, Smith and Hansen developed the SCOREM algorithm combining this statistic with graph theory (*SCOREM: statistical consolidation of redundant expression measures*, Nucleic Acids Research, 2011). This implementation is very much based on the SCOREM algorithm. The main changes are (1) the

current implementation is more generic, applicable to native R data structures, therefore able to be applied in other scenario than microarray analysis (2) it takes not-annotated features into account as well and (3) it is possible to directly calculate summary statistics from sub-groups.

Examples

```
## use a mock example
emat <- matrix(c(2,3,5,
                 8,9,2,
                 3,4,7,
                 0,2,1,
                 NA, 3, 1.2,
                 5, -3,4,
                 5,7,11), ncol=3, byrow=TRUE,
               dimnames=list(paste("row", 1:7, sep=""),NULL))
efac <- factor(c("a", "b", "c", NA, "b", "a", "a"),
               levels=letters[1:5])

print(emat)
kendallWmat(emat, efac, summary="none")
kendallWmat(emat, efac, summary="none", na.rm=FALSE)
kendallWmat(emat, efac, summary="mean")
kendallWmat(emat, efac, summary="mean", na.rm=FALSE)
kendallWmat(emat, efac, summary="median")
kendallWmat(emat, efac, summary="median", na.rm=FALSE)
kendallWmat(emat, efac, summary="max.mean.sig")
kendallWmat(emat, efac, summary="max.mean.sig", na.rm=FALSE)
kendallWmat(emat, efac, summary="max.var.sig")
kendallWmat(emat, efac, summary="max.var.sig", na.rm=TRUE)

## kendallW acts as an interface to matrix
kendallW(emat, efac, summary="none")

## kendallW acts as an interface to ExpressionSet
data(ribios.ExpressionSet, package="ribiosExpression")
kendallW(ribios.ExpressionSet,
  Biobase::fData(ribios.ExpressionSet)$GeneID,
  summary="none")
kendallW(ribios.ExpressionSet,
  Biobase::fData(ribios.ExpressionSet)$GeneID,
  summary="mean")
```

`kmeansGeneset` *Cluster gene-sets by enrichment profiles with k-means clustering, and select representative gene-sets by gene-set composition*

Description

Cluster gene-sets by enrichment profiles with k-means clustering, and select representative gene-sets by gene-set composition

Usage

```
kmeansGeneset(
  enrichProfMatrix,
  genesetGenes,
  optK = pmin(25, floor(nrow(enrichProfMatrix)/2)),
  iter.max = 15,
  nstart = 50,
  thrCumJaccardIndex = 0.5,
  maxRepPerCluster = 10,
  metaClusterColumns = 1:ncol(enrichProfMatrix)
)
```

Arguments

`enrichProfMatrix`
: A numeric matrix representing gene-set enrichment profile. Each row represent one gene-set and each column represent one enrichment profile, for instance a contrast in differential gene expression analysis. The values of the matrix represent enrichment of gene-sets, for instance enrichment score or absolute log10-transform p-values can be used. The row names are gene-set names.

`genesetGenes`
: A list of character strings, each element being genes of a gene-set in the `enrichProfMatrix`. The names of the list must exactly match the row-names of `enrichProfMatrix`, namely the names of gene-sets in the same order.

`optK`
: Integer, the number of initial clusters of gene-sets. Because one or more gene-sets may be selected from each gene-set cluster, the number of finally selected gene-sets is equal to or larger than `optK`.

`iter.max`
: Integer, the maximum numbers of iterations allowed. This parameter is passed to `kmeans`.

`nstart`
: Integer, how many random sets should be chosen to initialize cluster centers. This parameter is passed to `kmeans`.

`thrCumJaccardIndex`
: Numeric, between 0 and 1, the threshold of cumulative Jaccard Index. The larger the value is, the more gene-sets will be selected from each cluster

`maxRepPerCluster`
: Integer, maximum number of representative genesets per cluster. If NULL or NA, no limit is set.

`metaClusterColumns`
: Columns used to cluster the clusters by their average enrichment profile. By default, all columns are used.

This function performs `k-means` clustering of enrichment profiles of gene-sets. Within each cluster, we first identify the union set of unique genes covered any gene-set in the cluster, and then calculate Jaccard Index between genes in each gene-set and the union set. Gene-sets are sorted descendingly by the Jaccard Index, and the cumulative Jaccard Index is calculated. Among the sorted gene-sets, the gene-sets up to the position when the cumulative Jaccard Index exceeds `thrCumJaccardIndex` are selected (excluding redundant gene-sets).

The geneset clusters are ordered by their average profiles - similar clusters are near to each other.

Value

A list:

- kmeans Result object returned by `kmeans`.
- genesetClusterData A `data.frame` with following columns: `GenesetCluster`, `GenesetInd`, `GenesetName`, `JaccardIndex`, `CumJaccardIndex`, `IsRepresentative`.
- repGenesets Character vector, gene-set names that are selected as representative gene-sets from each gene-set clsuter.
- gsCompOverlapSelInd Factor vector, indicating the gene-set clusters represented by each representative gene-set.

Examples

```
set.seed(1887)
profMat <- matrix(rnorm(100), nrow=20,
    dimnames=list(sprintf("geneset%d", 1:20), sprintf("contrast%d", 1:5)))
gsGenes <- lapply(1:nrow(profMat), function(x)
    unique(sample(LETTERS, 10, replace=TRUE)))
names(gsGenes) <- rownames(profMat)
kmeansGeneset(profMat, gsGenes, optK=5)
```

---

list2mat *Convert a one-level list into an adjacency matrix*

Description

First-level list must have vectors of basic data types defined by R such as `characater`, `integer`, `number`, and `logical`.The function transforms such a list into adjacency matrix, rows of which are vector elements and columns of which are names of the list.

Usage

```
list2mat(list)
```

Arguments

`list`	A one-level list. See details

Value

An adjacency matrix. Row and column names are defined by unique elements and list names, respectively.

Author(s)

Jitao David Zhang <jitao_david.zhang@roche.com>

Examples

```
testList <- list(HSV=c("Adler", "Westermann", "Jansen"), FCB=c("Robben",
"Jansen", "Neuer"), S04=c("Westermann", "Neuer"))
list2mat(testList)

testList2 <- list(c("A", "B", "C"), c("B", "C", "D"), c("D", "E", "F"))
list2mat(testList2)

testList3 <- list(Worker1=0:8L, Worker2=5:13L, Worker3=8:16L, Worker4=16:24L)
list2mat(testList3)
```

---

logFCgage *Perform the GAGE analysis for EdgeResult and GmtList*

Description

Perform the GAGE analysis for EdgeResult and GmtList

Usage

```
logFCgage(edgeResult, gmtList)
```

Arguments

edgeResult	An EdgeResult object.
gmtList	A GmtList object.

Value

A data.frame containing enrichment analysis results.

`mergeCameraResults` *Merge CAMERA results using limma default parameters and biosCamera parameters*

Description

Merge CAMERA results using limma default parameters and biosCamera parameters

Usage

```
mergeCameraResults(
  matrix,
  index,
  designMatrix,
  contrast,
  featureLabels,
  weights = NULL,
  use.ranks = FALSE
)
```

Arguments

`matrix`	A numeric matrix, passed to `camera` and `biosCamera`
`index`	An index vector or a list of index vectors of features.
`designMatrix`	Design matrix.
`contrast`	A numeric vector of the same length as the number of columns in the design matrix, coefficients of contrasts.
`featureLabels`	A character vector of the same length as the number of rows of the matrix, feature labels, for instance gene symbols.
`weights`	NULL or numeric matrix of precision weights, passed to `camera`.
`use.ranks`	Logical, passed to `camera`.
	The function merges the output of `camera` with default options with the output of `biosCamera`, which appends additional information to camera methods, to return a comprehensive table as output.

Value

A `data.frame` containing merged CAMERA results.

See Also

`camera`, `biosCamera`

Examples

```
y <- matrix(rnorm(1000*6),1000,6)
features <- sprintf("Feature%d", 1:nrow(y))
design <- cbind(Intercept=1,Group=c(0,0,0,1,1,1))
# First set of 20 genes are genuinely deferentially expressed
index1 <- 1:20
y[index1,4:6] <- y[index1,4:6]+1
# The second set of 20 genes are not
index2 <- 21:40
index1Res <- mergeCameraResults(y, index=index1,
  designMatrix=design, contrast=c(0,1), featureLabels=features)
index1ListRes <- mergeCameraResults(y, index=list(index1),
   designMatrix=design, contrast=c(0,1), featureLabels=features)
index12ListRes <- mergeCameraResults(y, index=list(index1, index2),
   designMatrix=design, contrast=c(0,1), featureLabels=features)
```

---

minFDRvalue *Return the minimal FDR value from a FisherResultList*

Description

Return the minimal FDR value from a FisherResultList

Usage

```
minFDRvalue(object)
```

Arguments

object	A FisherResultList object

Value

A numeric value

---

minPvalue *Return the minimal p-value from a FisherResultList*

Description

Return the minimal p-value from a FisherResultList

Usage

```
minPvalue(object)
```

Arguments

object	A FisherResultList object

Value

A numeric value

`myGage` *Wrap the gage::gage method to report consistent results as the CAMERA method*

Description

Wrap the gage::gage method to report consistent results as the CAMERA method

Usage

```
myGage(logFC, gmtList, ...)
```

Arguments

logFC	A named vector of logFC values of genes
gmtList	A GmtList object containing gene-sets
...	Other parameters passed to gage

Value

A data.frame containing enrichment analysis results.

`orderByNumberInStr` *Order strings by numbers in them*

Description

Order strings by numbers in them

Usage

```
orderByNumberInStr(str, ...)
```

Arguments

str	A vector of character trings
...	Passed to order, by default decreasing is TRUE, i.e. the descending order is reported.

Value

An integer vector of indices.

See Also

factorByNumberInStr, which makes factors with levels ordered by numbers in the string

Examples

```
orderByNumberInStr(c("D1", "D10", "D15", "D3.5"))
```

---

parseCameraContributingGenes

Parse contributing genes by genesets from the result data.frame of the CAMERA method

---

Description

Parse contributing genes by genesets from the result data.frame of the CAMERA method

Usage

```
parseCameraContributingGenes(cameraResTbl, genesets)
```

Arguments

cameraResTbl	A tibble or data.frame holding results of CAMERA
genesets	Character strings, geneset labels

Value

A list of gene symbols, indexed by geneset names that are found in the results.

---

parseContributingGenes

Parse contributing genes from the CAMERA output file

Description

Parse contributing genes from the CAMERA output file

Usage

```
parseContributingGenes(str)
```

Arguments

str	Character string, containing contributing genes

Value

A list of data.frames, each containing two columns, Gene and Stat

Examples

```
parseContributingGenes("AKR1C4(-1.25), AKR1D1(-1.11)")
parseContributingGenes(c("AKR1C4(-1.25), AKR1D1(-1.11)",
                         "AKT1(1.24), AKT2(1.11), AKT3(1.05)"))
```

---

parseGenesetsContributingGenes

Parse contributing genes by genesets

Description

Parse contributing genes by genesets

Usage

```
parseGenesetsContributingGenes(str, genesets)
```

Arguments

str	Character strings, containing contributing genes
genesets	Character strings, geneset labels. Its length must match the length of str

Value

A data.frame containing genesets, genes, and statistics

Examples

```
parseGenesetsContributingGenes("AKR1C4(-1.25), AKR1D1(-1.11)", "Metabolism")
parseGenesetsContributingGenes(c("AKR1C4(-1.25), AKR1D1(-1.11)",
                         "AKT1(1.24), AKT2(1.11), AKT3(1.05)"),
                         c("Metabolism", "AKTs"))
```

parseGSEAdir *Parse an output directory of the Broad GSEA tool*

Description

Parse an output directory of the Broad GSEA tool

Usage

```
parseGSEAdir(dir)
```

Arguments

dir	Character string, path to output directory

Value

An AnnoBroadGseaRes object

prettyRonetGenesetNames *Pretty RONET Gene-set Names*

Description

Pretty RONET Gene-set Names

Usage

```
prettyRonetGenesetNames(x, nchar = 50)
```

Arguments

x	Character strings, RONET gene-set names
nchar	Integer, number of chararacters to be displayed.

Value

Character strings

Examples

```
strs <- c("ARNT_GeneID405_negativeTargets",
  "Neurophysiological_process_nNOS_signaling_in_neuronal_synapses",
  "NR5A1_GeneID2516_allTargets",
  "IL4_GeneID3565_negativeTargets",
  "Apoptosis_REACTOME")
prettyRonetGenesetNames(strs)
```

print.FisherResult *Print a FisherResult object*

Description

Print a FisherResult object

Usage

```
## S3 method for class 'FisherResult'
print(x, ...)
```

Arguments

x	A FisherResult object
...	Not used

Value

x, invisibly.

print.FisherResultList *Print a FisherResultList object*

Description

Print a FisherResultList object

Usage

```
## S3 method for class 'FisherResultList'
print(x, ...)
```

Arguments

x	A FisherResultList object
...	Not used

Value

x, invisibly.

print.FishersMethodResult *Print S3 object FishersMethodResult*

Description

Print S3 object FishersMethodResult

Usage

```
## S3 method for class 'FishersMethodResult'
print(x, ...)
```

Arguments

x	An object of the FishersMethodResult (S3) class
...	Not used

Value

x, invisibly.

printContributingGenes *Print contributing genes*

Description

Print contributing genes

Usage

```
printContributingGenes(geneLabels, geneValues)
```

Arguments

geneLabels	A vector of character strings
geneValues	A vector of numeric values

Value

A vector of character strings

pValue *Return P-values*

Description

Return P-values

Usage

```
pValue(object, ...)

## S4 method for signature 'FisherResult'
pValue(object)

## S4 method for signature 'FisherResultList'
pValue(object, ind, ...)

## S4 method for signature 'FisherResult'
fdrValue(object)

## S4 method for signature 'FisherResultList'
fdrValue(object, ind, ...)
```

Arguments

object	An object
...	Other parameters
ind	An integer or logical vector for subsetting

Value

A numeric vector of p-values.

Methods (by class)

- pValue(FisherResult): Return the p-value from a FisherResult
- pValue(FisherResultList): Return the p-values from a FisherResultList. If ind is missing, all p-values are returned; otherwise, the subset indicated by ind is returned.
- fdrValue(FisherResult): Return the FDR-value from a FisherResult
- fdrValue(FisherResultList): Return the FDR-values from a FisherResultList. If ind is missing, all FDR-values are returned; otherwise, the subset indicated by ind is returned.

readCameraResults *Read CAMERA results into a tibble object*

Description

Read CAMERA results into a tibble object

Usage

```
readCameraResults(file, minNGenes = 3, maxNGenes = 1000)
```

Arguments

file	CAMERA results file
minNGenes	NULL or integer, genesets with fewer genes are filtered out
maxNGenes	NULL or integer, genesets with more genes are filtered out

Value

A tibble containing the CAMERA results.

readDefaultGenesets *Read default genesets for gene-set enrichment analysis*

Description

In Roche Bioinformatics we use a default collection of gene-sets for gene-set enrichment analysis. This function loads this collection.

Usage

```
readDefaultGenesets(path, mps = FALSE)
```

Arguments

path	Character, path to the directory where the gmt files are stored
mps	Logical, whether molecular-phenotypic screening (MPS) genesets should be read in as pathway-centric namespaces (TRUE) or as one namespace named MolecularPhenotyping (FALSE).

Details

The default collection includes both publicly available genesets as well as proprietary genesets, and therefore they are not included as part of the ribios package.

Publicly available genesets include

- MSigDB: collections C2, C7 and Hallmark
- RONET: which is a collection of publicly available pathway databases including REACTOME and NCI-Nature
- goslim

Value

A `GmtList` object containing the default gene-set collections.

Examples

```
## Not run:
  ## this cannot be run because the files are not located there
  ## readDefaultGenesets("/tmp/defaultGmts")

## End(Not run)
```

`readMPSGmt` *Read molecular-phenotyping genesets*

Description

Read molecular-phenotyping genesets

Usage

```
readMPSGmt(file)
```

Arguments

`file`	GMT file which stores default molecular-phenotyping genesets

Value

A `GmtList` object containing molecular-phenotypic screening (MPS) categories and genes

readRonetGmt *Read RONET GMT files with namespace information*

Description

Read RONET GMT files with namespace information

Usage

```
readRonetGmt(file)
```

Arguments

file	A GMT file in the RONET format, where in the 'desc' field a namespace is appended at the beginning, separated from the rest of the description with a pipe

Value

A `GmtList` object with an additional 'namespace' item in each list

readSigCameraResults *Read significant CAMERA results into a tibble*

Description

Read significant CAMERA results into a tibble

Usage

```
readSigCameraResults(
  file,
  returnAllContrasts = TRUE,
  maxPValue = 0.01,
  minAbsEffectSize = 0.5,
  minNGenes = 5,
  maxNGenes = 200,
 excludeNamespace = c("goslim", "immunespace", "immunomics", "mbdisease", "mbpathology",
    "mbtoxicity", "msigdbC7", "msigdbC2", "MolecularPhenotyping")
)
```

Arguments

file	A tsv file, output of biosCamera
returnAllContrasts	Logical, if TRUE, results of all contrasts for gene-sets that are significant in at least one contrast are returned.
maxPValue	Numeric, max unadjusted P-value of CAMERA that is considered significant
minAbsEffectSize	Numeric, minimal absolute effect size
minNGenes	Integer, size of the smallest gene set that is considered
maxNGenes	Integer, size of the largest gene set that is considered
excludeNamespace	Character, vector of namespaces to be excluded

Value

A tibble containing filtered CAMERA results.

---

readSigCameraScoreMatrix

Read significant CAMERA results into a matrix

Description

Read significant CAMERA results into a matrix

Usage

```
readSigCameraScoreMatrix(file, ...)
```

Arguments

file	A tsv file, output of biosCamera
...	passed to readSigCameraResults

Value

A numeric matrix with gene-sets as rows and contrasts as columns.

removeFromGeMS *Rmove one or gene sets of the same source and user from GeMS*

Description

Rmove one or gene sets of the same source and user from GeMS

Usage

```
removeFromGeMS(
  setName = "",
  source = "",
  user = ribiosUtils::whoami(),
  subtype = ""
)
```

Arguments

setName	A vector of character strings, defining set names to be renamed. They must all have the same source, user, and subtype
source	Character string, source of the gene set(s)
user	Character string, user name
subtype	Character string, subtype of the gene set(s)

Value

Response code or error message returned by the GeMS API. A value of 200 indicates a successful insertion.

See Also

insertGmtListToGeMS

Examples

```
## Not run:
  testList <- list(list(name="GS_A", desc=NULL, genes=c("MAPK14", "JAK1", "EGFR")),
    list(name="GS_B", desc="gene set B", genes=c("ABCA1", "DDR1", "DDR2")),
    list(name="GS_C", desc="gene set C", genes=NULL))
  testGmt <- BioQC::GmtList(testList)
  ## insertGmtListToGeMS(testGmt, geneFormat=0, source="Test")
  ## removeFromGeMS(setName=c("GS_A", "GS_B", "GS_C"), source="Test")

## End(Not run)
```

removeFromGeMSBody *Message body to remove one or gene sets of the same source and user from GeMS*

Description

Message body to remove one or gene sets of the same source and user from GeMS

Usage

```
removeFromGeMSBody(
  setName = "",
  source = "",
  user = ribiosUtils::whoami(),
  subtype = ""
)
```

Arguments

setName	A vector of character strings, defining set names to be renamed. They must all have the same source, user, and subtype
source	Character string, source of the gene set(s)
user	Character string, user name
subtype	Character string, subtype of the gene set(s)

Value

A list of genesets to be removed, to be sent as message body

Examples

```
removeFromGeMSBody(setName=c("GS_A", "GS_B", "GS_C"), source="Test")
```

ronetGeneSetNamespace *Extract gene-set namespace from RONET GMT files*

Description

Extract gene-set namespace from RONET GMT files

Usage

```
ronetGeneSetNamespace(gmtList)
```

Arguments

gmtList	A GmtList object read from a RONET GMT file

Value

Character vector of the same length, indicating categorie

---

show,AnnoBroadGseaRes-method *Show a anonBroadGseaRes object*

Description

Show a anonBroadGseaRes object

Usage

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

Arguments

object	A AnnoBroadGseaRes object export

---

show,AnnoBroadGseaResItem-method *Show an AnnoBroadGseaResItem object*

Description

Show an AnnoBroadGseaResItem object

Usage

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

Arguments

object	An annoBroadGseaResItem object export

`show,BroadGseaResItem-method` *Show a BroadGseaResItem object*

Description

Show a BroadGseaResItem object

Usage

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

Arguments

`object`	A BroadGseaResItem object export

`sigGeneSet` *Return names of gene-sets that are significantly enriched given the FDR threshold*

Description

Return names of gene-sets that are significantly enriched given the FDR threshold

Usage

```
sigGeneSet(object, fdr)
```

Arguments

`object`	A FisherResultList object
`fdr`	Numeric, FDR value threshold

Value

A character vector

sigGeneSetTable *Return a data.frame of significantly enriched gene-sets*

Description

Return a data.frame of significantly enriched gene-sets

Usage

```
sigGeneSetTable(object, fdr)
```

Arguments

object	A FisherResultList object
fdr	Numeric, FDR value threshold

Value

A data.frame

topGeneSetTable *Return a data.frame of top gene-sets with the lowest p-values*

Description

Return a data.frame of top gene-sets with the lowest p-values

Usage

```
topGeneSetTable(object, N)
```

Arguments

object	An FisherResultList object
N	Integer, the number of returned gene-sets

Value

A data.frame

topOrSigGeneSetTable *Return a data.frame of significantly enriched gene-sets with a minimum number*

Description

Return a data.frame of significantly enriched gene-sets with a minimum number

Usage

```
topOrSigGeneSetTable(object, fdr = 0.05, N = 10)
```

Arguments

object	An FisherResultList object
fdr	Numeric, the treshold of FDR value
N	Integer, the number of returned gene-sets The total number of returned gene-sets are determined by the maximum of N and the counts of gene-sets that have FDR lower than fdr.

Value

A data.frame.

writeGmt *Write an GmtList object into a file*

Description

Write an GmtList object into a file

Usage

```
writeGmt(gmtList, file)
```

Arguments

gmtList	A GmtList object
file	Character string, output file name

Value

Invisibly returns NULL. Called for its side effect of writing the GMT file.

Note

The function will be moved to BioQC once the ribiosIO is reposited in CRAN

Examples

```
gmtFile <- system.file("extdata", "example.gmt", package="ribiosGSEA")
mySet <- BioQC::readGmt(gmtFile)[1:5]
myTempFile <- tempfile()
writeGmt(mySet, file=myTempFile)
readLines(myTempFile)
```

---

zscoreDGE *Calculate mid-p quantile residuals*

Description

Calculate mid-p quantile residuals

Usage

```
zscoreDGE(y, design = NULL, contrast = ncol(design))
```

Arguments

y	An DGEList object
design	Design matrix
contrast	Contrast vector
	The function is a carbon copy of edgeR:::.zscoreDGE, which is unfortunately not exported

Value

A numeric matrix of z-scores with the same dimensions as the input count matrix.

Examples

```
dgeMatrix <- matrix(rpois(1200, 10), nrow=200)
dgeList <- DGEList(dgeMatrix)
dgeList <- edgeR::estimateCommonDisp(dgeList)
dgeDesign <- model.matrix(~gl(2,3))
dgeZscore <- zscoreDGE(dgeList, dgeDesign, contrast=c(0,1))
head(dgeZscore)
```

Index