Package 'mdgsa'

March 30, 2021

Type Package

Title Multi Dimensional Gene Set Analysis.

Version 1.22.0

Date 2013-09-26

Author David Montaner <dmontaner@cipf.es>

Maintainer David Montaner <dmontaner@cipf.es>

URL https://github.com/dmontaner/mdgsa, http://www.dmontaner.com

Description Functions to preform a Gene Set Analysis in several genomic dimensions. Including methods for miRNAs.

License GPL

LazyLoad yes

LazyData FALSE

Depends R (>= 2.14)

Imports AnnotationDbi, DBI, GO.db, KEGG.db, cluster, Matrix

Suggests BiocStyle, knitr, rmarkdown, limma, ALL, hgu95av2.db, RUnit, BiocGenerics

VignetteBuilder knitr

biocViews GeneSetEnrichment, Annotation, Pathways, GO

git_url https://git.bioconductor.org/packages/mdgsa

git_branch RELEASE_3_12

git_last_commit fcb676e

git_last_commit_date 2020-10-27

Date/Publication 2021-03-29

R topics documented:

annotFilter 2
annotList2mat 3
annotMat2list 4
getGOnames 5
getKEGGnames 5
getOntology 6
goLeaves 7

indexTransform 8
mdGsa 9
mdPat 10
plotMdGsa 12
propagateGO 13
pval2index 14
revList 15
splitOntologies 16
transferIndex 17
uvGsa 18
uvPat 20
uvSignif 21

Index **22**

`annotFilter` *Checks and filters an annotation list.*

Description

Checks that the annotated genes (those in the annotation list) are consistent with the universe of genes defined by the ranking index. Filters out functional blocks too 'big' or too 'small'.

Usage

```
annotFilter(annot, index, minBlockSize = 10, maxBlockSize = 500,
  verbose = TRUE)
```

Arguments

`annot`	an annotation list.
`index`	ranking index. Vector, matrix or data.frame
`minBlockSize`	minimum block size kept
`maxBlockSize`	maximum block size kept
`verbose`	verbose

Details

`index` is optional. When it is not provided, the annotation lists is just filtered out by the sizes of the blocks of genes defined in the list.

If a ranking index is provided its names are assumed to be the universe of genes under study. The genes in the annotation list are compared against those of the ranking index and the ones not belonging to the universe are removed out form the annotation in order to compute the size of each functional block. Then the list is filtered by sizes; too big and too small blocks are removed.

No transformation is done over the ranking index or its names (gene IDs).

`index` may just be a character vector containing the names of the genes in the universe, that is, the names or row names of the ranking index.

Value

a filtered annotation list.

Author(s)

David Montaner <dmontaner@cipf.es>

See Also

annotMat2list

Examples

```
rindex <- 1:10
names (rindex) <- paste ("gene", rindex, sep = "")
rindex

annot <- list (paste ("gene", 1:2, sep = ""),          ##too small block
               paste ("gene", 1:6, sep = ""),          ##right size
               paste ("gene", 1:5, sep = ""),          ##too big block
               paste ("gene", c(1:3, 1:3), sep = ""))  ##duplicated IDs
annot[[2]][1] <- NA
annot[[2]][2] <- ""
annot[[2]][3] <- "BAD_ID"
annot

annotFilter (annot, minBlockSize = 3, maxBlockSize = 5)
annotFilter (annot, rindex, minBlockSize = 3, maxBlockSize = 5)
```

---

annotList2mat *Convert an annotation list into an annotation matrix.*

Description

Converts an annotation list to an annotation matrix. The annotation matrix should have 2 columns, the first one with the gene ids; the second one with the annotation ids.

Usage

```
annotList2mat(lis, tag = "listPos")
```

Arguments

lis	annotation list.
tag	substitutes missing list names if any.

Details

Each element of the annotation list represents a functional block; it is a character vector containing the gene ids annotated under the function. The names of the list are the annotation ids.

Value

An annotation matrix: the first column contains the gene or feature ids, the second column contains the Gene Set or functional block ids.

Author(s)

David Montaner <dmontaner@cipf.es>

See Also

annotMat2list, revList, split

Examples

```
lis <- list (Block1 = c("gen1", "gen2"), Block2 = c("gen3"))
annotList2mat (lis)
```

---

annotMat2list *Convert an annotation matrix into an annotation list.*

Description

Converts an annotation matrix to an annotation list. The annotation matrix should have 2 columns, the first one with the gene ids; the second one with the annotation ids.

Usage

```
annotMat2list(mat)
```

Arguments

mat	annotation matrix; gene IDs in the first column; block IDs in the second column.

Details

Each element of the annotation list represents a functional block; it is a character vector containing the gene ids annotated under the function. The names of the list are the annotation ids.

Value

An annotation list: elements of the list are vectors of genes; names of the list are Gene Set ids.

Author(s)

David Montaner <dmontaner@cipf.es>

See Also

annotList2mat, revList, split

Examples

```
mat <- cbind (c("gen1", "gen2", "gen3"), c("Block1", "Block1", "Block2"))
annotMat2list (mat)
```

getGOnames *Get Gene Ontology names*

Description

Finds the GO name form GO id.

Usage

```
getGOnames(x, verbose = TRUE)
```

Arguments

x	a character vector of GO ids.
verbose	verbose.

Details

Uses the library GO.db.

x may be a data.frame. In such case, GO ids are expected in its row names.

Value

A character vector with the corresponding GO names.

Author(s)

David Montaner <dmontaner@cipf.es>

See Also

propagateGO, goLeaves, splitOntologies, getKEGGnames, getOntology

Examples

```
getGOnames (c("GO:0000018", "GO:0000038", "BAD_GO"))
```

getKEGGnames *Get KEGG names*

Description

Finds the KEGG name form KEGG id.

Usage

```
getKEGGnames(x, verbose = TRUE)
```

Arguments

x	a character vector of KEGG ids.
verbose	verbose.

Details

Uses the library KEGG.db.

x may be a data.frame. In such case, GO ids are expected in its row names.

Value

A character vector with the corresponding KEGG names.

Author(s)

David Montaner <dmontaner@cipf.es>

See Also

getGOnames

Examples

```
getKEGGnames (c("00010", "00020", "BAD_KEGG"))
```

---

getOntology *Get GO term Ontology*

Description

Finds the ontology of a term from its id.

Usage

```
getOntology(x, verbose = TRUE)
```

Arguments

x	a character vector of GO ids.
verbose	verbose.

Details

Uses the library GO.db.

x may be a data.frame. In such case, GO ids are expected in its row names.

Value

A character vector with the corresponding GO names.

Author(s)

David Montaner <dmontaner@cipf.es>

See Also

getGOnames, propagateGO, goLeaves, splitOntologies, getKEGGnames

Examples

```
getOntology (c("GO:0000018", "GO:0005788", "BAD_GO"))
```

---

goLeaves *Keep just leaf nodes from the Gene Ontology DAG.*

Description

Cuts significant terms and filters out all redundant GO terms from a list of uvGsa results.

Usage

```
goLeaves(gsaout, cutoff = 0.05, pvalue = "padj", statistic = "lor",
  verbose = TRUE, sort = TRUE)
```

Arguments

gsaout	data.frame; output from uvGsa.
cutoff	p-value cutoff for considering significant a Gene Set.
pvalue	p-value column to be used. Default is named "padj" as in uvGsa output.
statistic	name of the column containing the log odds ratio from the uvGsa analysis.
verbose	verbose
sort	if TRUE the output data.frame is ordered according to significance.

Details

Uses the library GO.db to find the 'ancestors' of each GO term. Those ancestors are discarded form the uvGsa results.

Alternatively, the function may also take a character vector of GO ids in the gsaout parameter. In such case the function returns also a character vector of GO ids, containing just the GO terms being "leaves" of the original set.

Value

The input data.frame but keeping just the 'significant' and 'non redundant' GO terms.

Author(s)

David Montaner <dmontaner@cipf.es>

See Also

uvGsa, uvPat, propagateGO, pval2index

Examples

```
getGOnames (c ("GO:0006259", "GO:0006915", "GO:0043280"))
goLeaves   (c ("GO:0006259", "GO:0006915", "GO:0043280"))

## Not run:
res <- uvGsa (rindex, goAnnotList)
goLeaves (res)

## End(Not run)
```

---

indexTransform *Transform ranking index distribution.*

Description

The function performs a transformation of the ranking index so that its distribution is suitable as independent variable of the logistic regression model.

Usage

```
indexTransform(index, method = "normalize")
```

Arguments

index	a ranking index; a numeric vector, matrix or data.frame.
method	transformation method. See details.

Details

Works for vector, matrices and data.frames. In the case of matrices the function transforms each column separately from the other ones.

Two methods are currently implemented:

- **normalize:** transforms the index into quantiles of a normal distribution.
- **standardize:** performs an statistical standardization by subtracting the mean and dividing by the standard deviation.

Value

A transformed index. Its class will be that of the the input object.

Author(s)

David Montaner <dmontaner@cipf.es>

See Also

transferIndex, uvGsa, mdPat

Examples

```
myIndex <- runif (1000)
myTransformedIndex <- indexTransform (myIndex)
plot (myIndex, myTransformedIndex)
```

mdGsa *Multi-Dimensional Gene Set Analysis.*

Description

Performs a Multi-Variate Gene Set Analysis for two genomic measurements.

Usage

```
mdGsa(index, annot, p.adjust.method = "BY", family = quasibinomial(),
  verbose = TRUE, verbosity = 100, fulltable = FALSE,
  useColnames = TRUE, ...)
```

Arguments

index	ranking index, generally a two column matrix.
annot	an annotation list.
p.adjust.method	p-value adjustment method for multiple testing.
family	see glm.
verbose	verbose.
verbosity	integer indicating which iterations should be indicated when verbose = TRUE.
fulltable	if TRUE, 'sd', 't' and 'convergence' indicator from the glm fit are included in the output.
useColnames	if TRUE the names of the two first columns of the matrix 'index' are used in the results data.frame.
...	further arguments to be pasted to glm.fit, for instance 'weights'.

Details

'index' must be a numerical matrix or data.frame with at least two columns.

If there are more than three columns, the ranking indexes are taken form the two first one. The remaining columns are used as covariates to correct for within the analysis.

Default p-value correction is "BY".

In the output data.frame there are three parameters of each type: 'lor', 'pval', ... one for each of the two genomic conditions analyzed and the third one for the interaction between them.

If available, names of the fist two columns of the index matrix are used in the output data.frame. Changing the order of these two first columns will change the report order, but will not change the interpretation of the results. See Montaner et al. (2010) for further details on the algorithm.

Value

A data.frame with a row for each Gene Set or block. Columns are:

N: number of genes annotated to the Gene Set.

lor: log Odds Ratio estimated for the Gene Set.

pval: p-values associated to each log Odds Ratio.

`padj`: adjusted p-values.

`sd`: standard deviations associated to each log Odds Ratio.

`t`: t statistic associated to each log Odds Ratio.

Apart from the 'N' coefficient, all other indices appear in triplicate: one coefficient for each genomic condition and a third one for the interaction.

Author(s)

David Montaner <dmontaner@cipf.es>

References

Montaner et al. (2010) "Multidimensional Gene Set Analysis of Genomic Data." PLoS ONE.

See Also

uvGsa, mdPat, glm.fit, p.adjust

Examples

```
rindexMat <- matrix (rnorm (2000), ncol = 2)
colnames (rindexMat) <- c ("genomicVar1", "genomicVar2")
rownames (rindexMat) <- paste0 ("gen", 1:1000)

annotList <- list (geneSet1 = sample (rownames (rindexMat), size = 10),
                   geneSet2 = sample (rownames (rindexMat), size = 15),
                   geneSet3 = sample (rownames (rindexMat), size = 20))

res <- mdGsa (rindexMat, annotList)
res
```

---

mdPat *Multi-Dimensional Gene Set Analysis Pattern Classification.*

Description

Classifies significant patterns form a Multi-Variate Gene Set Analysis.

Usage

```
mdPat(gsaout, cutoff = 0.05, pvalue = "padj")
```

Arguments

gsaout	data.frame; output from mdGsa.
cutoff	p-value cutoff for considering significant a Gene Set.
pvalue	p-value column to be used. Default is named "padj" as in mdGsa output.

Details

Sign of the three 'lor' and p-values are used to classify functional blocks. The classification is done in the two dimensional space previously analyzed by mdGsa.

All possible functional block classifications in the bi-dimensional gene set analysis are:

- q1i: block displaced toward quadrant **1** (0 < X & 0 < Y) with interaction.
- q2i: block displaced toward quadrant **2** (0 > X & 0 < Y) with interaction.
- q3i: block displaced toward quadrant **3** (0 > X & 0 > Y) with interaction.
- q4i: block displaced toward quadrant **4** (0 < X & 0 > Y) with interaction.
- q1f: block displaced toward quadrant **1**, no interaction.
- q2f: block displaced toward quadrant **2**, no interaction.
- q3f: block displaced toward quadrant **3**, no interaction.
- q4f: block displaced toward quadrant **4**, no interaction.
- xh: block shifted to **positive X** values.
- xl: block shifted to **negative X** values.
- yh: block shifted to **positive Y** values.
- yl: block shifted to **negative Y** values.
- b13: bimodal block. Half of the genes displaced towards quadrant **1** and the other half towards quadrant **3**.
- b24: bimodal block. Half of the genes displaced towards quadrant **2** and the other half towards quadrant **4**.
- NS: **non significant** block.

Value

A character vector indicating the pattern associated to each Gene Set.

Author(s)

David Montaner <dmontaner@cipf.es>

References

Montaner et al. (2010) "Multidimensional Gene Set Analysis of Genomic Data." PLoS ONE.

See Also

mdGsa, uvPat

Examples

```
N <- c (10, 20, 30, 40)
lor.X <- c (1.45, -0.32, 1.89, -1.66)
lor.Y <- c (2.36, -1.86, 0.43, -2.01)
lor.I <- c (0.89, -0.12, 0.24,  3.55)
pval.X <- c (0.001, 0.002, 0.003, 0.06)
pval.Y <- c (0.002, 0.003, 0.06,  0.07)
pval.I <- c (0.003, 0.02,  0.05,  0.08)
padj.X <- p.adjust (pval.X, "BY")
```

```
padj.Y <- p.adjust (pval.Y, "BY")
padj.I <- p.adjust (pval.I, "BY")

mdGsa.res <- as.data.frame (cbind (N,
                                   lor.X, lor.Y, lor.I,
                                   pval.X, pval.Y, pval.I,
                                   padj.X, padj.Y, padj.I))
mdGsa.res

mdGsa.res[,"pat"] <- mdPat (mdGsa.res)
mdGsa.res
```

`plotMdGsa` *Plot Multi-Dimensional Gene Set*

Description

Plots confidence region for a Gene Set in a two dimensional space.

Usage

```
plotMdGsa(index, block, cr = 0.95, pch = ".", pch.block = 20, lwd = 2,
  col.all = "blue", col.block = "red", project = FALSE,
  col.proj = "green", diagonals = FALSE, col.diag = "gray", ...)
```

Arguments

`index`	matrix or data frame with the two columns of ranking statistics.
`block`	matrix or data frame with **gene** ids in the first column and **gene set** ids in the second column.
`cr`	level of the confidence region.
`pch`	plotting character for all genes.
`pch.block`	plotting character for the genes in the gene set or functional block.
`lwd`	line width. Used when drawing ellipses and other lines.
`col.all`	color used to represent all genes.
`col.block`	color used to represent the genes in the gene set being plotted.
`project`	if TRUE projection over the axis are displayed for the genes of the gene set.
`col.proj`	color used to plot the projection.
`diagonals`	if TRUE diagonals are plotted.
`col.diag`	color used to plot the diagonals.
`...`	arguments to be passed to plot

Details

Black dots show all genes in the dataset. Red stars show the genes in the Gene Set. Blue axis show the "center" of the distribution of all genes; blue ellipse shows the confidence region for all genes. Red axis show the "center" of the distribution of the genes in the Gene Set; red ellipse shows the confidence region for genes in the Gene Set.

Value

A plot.

Author(s)

David Montaner <dmontaner@cipf.es>

See Also

mdGsa, mdPat, mdPat, ellipsoidPoints

Examples

```
## Not run:
res <- mdGsa (rindexMat, annotList)
plotMdGsa (rindexMat, block = annotList[["GO:0006915"]])

## End(Not run)
```

---

propagateGO *Propagate Gene Ontology annotation.*

Description

Genes annotated under a GO term inherit the annotation from all its ancestors.

Usage

```
propagateGO(annot, verbose = FALSE)
```

Arguments

annot	annotation list or matrix.
verbose	verbose

Details

Uses the library GO.db.

Value

A annotation matrix or list with the propagated annotation.

Author(s)

David Montaner <dmontaner@cipf.es>

See Also

annotMat2list, annotFilter

Examples

```
mat <- cbind (c("gene1", "gene2"), c("GO:0034390", "GO:0042889"))
mat
propagateGO (mat)

li <- list ('GO:0034390' = "gene1", 'GO:0042889' = "gene2")
li
propagateGO (li)
```

`pval2index` *Transform p-values in into a ranking index.*

Description

After a genomic test, p-values are numerical indexes which account for certain biological characteristic. By definition p-values are bounded between zero and one, but this may not be suitable as an index. Moreover, p-values are always derived form a statistic which sign may be important. The function helps transforming the p-value and its associated statistic into a ranking index.

Usage

```
pval2index(pval, sign, names = NULL, log = TRUE, offset, verbose = TRUE)
```

Arguments

`pval`	a vector or matrix of p-values.
`sign`	a vector or matrix of signs associated to the p-values.
`names`	a character vector of the names of the features.
`log`	= TRUE
`offset`	value used to replace p-values equal to zero
`verbose`	verbose

Details

The default transformation is (-1) * log (pval) * sign (sign). When log = FALSE the transformation is (1 - pval) * sign (sign).

If `sign` is missing all p-values are associated wit a positive sign.

Missing values are allowed and return NA values.

An `offset` may be provided to replace p-values equal to zero when `log = TRUE`. In such way **infinite** values are not generated. If the `offset` parameter is not provided, the minimum p-value other than zero is used for the replacement. You can explicitly specify `offset = 0` if you want `Inf` values to be returned.

By default the names of the output vector (or row names in a matrix) are those of `pval` or `sign`. If `names` is provided, then it is used instead.

Value

A transformed index. A vector or matrix, depending on the input parameters.

Author(s)

David Montaner <dmontaner@cipf.es>

See Also

indexTransform

Examples

```
my.statistic <- rnorm (1000)
my.pvalue <- 2 * pnorm (my.statistic)
my.pvalue[my.pvalue > 1] <- 2 - my.pvalue[my.pvalue > 1]

index <- pval2index (pval = my.pvalue, sign = my.statistic)

#par (mfrow = c (1,2))
#plot (my.statistic, my.pvalue)
#plot (my.statistic, index)


## Zero p-values
p <- c (0:10)/10
p
pval2index (p)
pval2index (p, offset = 0)
pval2index (p, offset = 0.000001)

## Missing p-values
p <- c(0:10, NA)/10
p
pval2index (p)
pval2index (p, offset = 0)
pval2index (p, offset = 0.000001)
pval2index (p, log = FALSE)
pval2index (p, offset = 0, log = FALSE)

## Matrix
p <- matrix (c(0:10, NA)/10, ncol = 3)
p
pval2index (p)
pval2index (p, offset = 0)
pval2index (p, offset = 0.000001)
pval2index (p, log = FALSE)
pval2index (p, offset = 0, log = FALSE)
```

---

revList *Revert an annotation list.*

Description

Inverts a list: names to elements / elements to names

Usage

```
revList(lis, tag = "listPos")
```

Arguments

lis	annotation list.
tag	substitutes missing list names if any.

Value

An inverted list.

Author(s)

David Montaner <dmontaner@cipf.es>

See Also

annotMat2list, revList annotMat2list, annotList2mat

Examples

```
lis <- list (Block1 = c("gen1", "gen2"), Block2 = c("gen1", "gen3"))
revList (lis)
```

---

splitOntologies *Split an annotation list of GO terms by ontologies.*

Description

Splits an annotation list of GO terms according to the ontology to which each term belongs to.

Usage

```
splitOntologies(annot, na.rm = TRUE, verbose = TRUE)
```

Arguments

annot	annotation list.
na.rm	if TRUE 'unknown' terms are excluded.
verbose	verbose

Details

Uses the information form the library GO.db. If some id could not be associated to any ontology, they are returned in an unknown ontology named "missing".

Value

A list with tree components, one for each ontology. A fourth component is included is some term could not be allocated to any of the three GO ontologies.

Author(s)

David Montaner <dmontaner@cipf.es>

See Also

propagateGO, goLeaves

Examples

```
getGOnames (c ("GO:0006915", "GO:0016020", "GO:0008152", "GO:0015288"))

annot <- list ("GO:0006915" = c ("g1"),
               "GO:0016020" = c ("g2", "g3"),
               "GO:0008152" = c ("g1", "g2", "g3"),
               "GO:0015288" = c ("g4", "g5"))
annot
splitOntologies (annot)
```

`transferIndex` *Transfer a ranking index from regulatory elements, such as miRNAs, to genes.*

Description

Transfers the ranking index information from some regulatory elements to their target genes. It can be for instance used to infer gene regulation levels from miRNA differential expression levels. Afterwords, the inferred gene index can be explored in a univariate gene set analysis using `uvGsa`, or in a multivariate gene set analysis using `mdGsa`.

Usage

```
transferIndex(index, targets, method = "sum", verbose = TRUE,
  transferMatrix = FALSE)
```

Arguments

`index`	ranking index, generally a numerical named vector.
`targets`	a list describing the regulation blocks. See details.
`method`	the method to collapse the gene information. Currently 'sum' and 'average' are available.
`verbose`	verbose.
`transferMatrix`	if true the transference matrix is returned instead of the ranking statistic.

Details

The function was in principle designed to transfer a ranking index defined for miRNAs to their target genes, but it may also be used to deal with some other regulatory elements as transcription factors for instance.

If we have for instance a t statistic accounting for miRNAs differential expression levels, we may add up (or average) all t values from the miRNAs that regulate a gene in order to derive an interference score for that gene. Thus we may get an interference index which may be interpreted in terms of functional blocks or gene sets.

`index` may be a matrix. In such case just the first column will be used.

`targets` should be a named list. The names of the list have to coincide (or at least overlap) with the names in the `index`. The elements of the list are character vectors containing gene IDs. In the miRNA example, each element of `targets` will contain the genes regulated by a miRNA, and the miRNA IDs will appear in the names of the `targets` list as well as in the ranking index.

Value

A ranking index transferred to the IDs in the elements of the list `targets`.

Author(s)

David Montaner <dmontaner@cipf.es>

See Also

uvGsa, mdGsa, indexTransform

Examples

```
## miRNA to gene list (targets)
targets <- list (mirna1 = "g1", mirna2 = c ("g1", "g2"), mirna3 = "g3")

## original index
index <- rnorm (5)
names (index) <- paste0 ("mirna", 1:5)

##transfered index
tindex <- transferIndex (index, targets)

##transformed (normalized) index
rindex <- indexTransform (tindex)

##NOTICE in this case:
index["mirna1"] + index["mirna2"] == tindex["g1"]

## Not run:
res <- uvgsa (rindex, annot)

## End(Not run)
```

---

uvGsa *Uni-Variate Gene Set Analysis.*

Description

Performs a Uni-Variate Gene Set Analysis using a logistic regression model.

Usage

```
uvGsa(index, annot, p.adjust.method = "BY", family = quasibinomial(),
  verbose = TRUE, verbosity = 100, fulltable = FALSE, ...)
```

Arguments

index	ranking index, generally a numerical named vector.
annot	an annotation list.
p.adjust.method	p-value adjustment method for multiple testing.
family	see glm.fit.
verbose	verbose.
verbosity	integer indicating which iterations should be indicated if verbose = TRUE.
fulltable	if TRUE, 'sd', 't' and 'convergence' indicator from the glm fit are included in the output.
...	further arguments to be pasted to glm.fit, for instance 'weights'.

Details

'index' may also be a numerical matrix or data.frame. If such a matrix has more than one column, the ranking index is taken form the first one. The remaining columns are used as covariates to correct for within the analysis.

Default p-value correction is "BY".

Value

A data.frame with a row for each Gene Set or block. Columns are:

N: number of genes annotated to the Gene Set.

lor: log Odds Ratio estimated for the Gene Set.

pval: p-values associated to each log Odds Ratio.

padj: adjusted p-values.

sd: standard deviations associated to each log Odds Ratio.

t: t statistic associated to each log Odds Ratio.

Author(s)

David Montaner <dmontaner@cipf.es>

See Also

mdGsa, uvPat, glm.fit, p.adjust

Examples

```
rindex <- rnorm (1000)
names (rindex) <- paste0 ("gen", 1:1000)

annotList <- list (geneSet1 = sample (names (rindex), size = 10),
                   geneSet2 = sample (names (rindex), size = 15),
                   geneSet3 = sample (names (rindex), size = 20))

res <- uvGsa (rindex, annotList)
res
```

uvPat *Uni-Variate Gene Set Analysis Pattern Classification.*

Description

Classifies significant patterns form a Uni-Variate Gene Set Analysis.

Usage

```
uvPat(gsaout, cutoff = 0.05, pvalue = "padj", statistic = "lor")
```

Arguments

gsaout	data.frame; output from uvGsa.
cutoff	p-value cutoff for considering significant a Gene Set.
pvalue	p-value column to be used. Default is named "padj" as in uvGsa output.
statistic	name of the column containing the log odds ratio from the uvGsa analysis.

Details

Sign of the 'lor' and p-value are used to define functional blocks as up-regulated, down-regulated or not enriched.

Value

A numeric vector (values: -1, 0, 1) indicating relationship between the Gene Set and the ranking variable:

1: indicates that the gene set is significantly associated to high values of the ranking statistic.

-1: indicates that the gene set is significantly associated to low values of the ranking statistic.

0: indicates that the gene set not related to the ranking statistic (no enrichment).

Author(s)

David Montaner <dmontaner@cipf.es>

See Also

uvGsa, mdPat

Examples

```
uvGsa.res <- as.data.frame (list (N    = c (10, 20, 30, 40),
                                  lor  = c (1.45, -0.32, 1.89, -1.66),
                                  pval = c (0.001, 0.002, 0.05, 0.06)))
uvGsa.res[,"padj"] <- p.adjust (uvGsa.res$pval, "BY")
uvGsa.res

uvGsa.res[,"pat"] <- uvPat (uvGsa.res)
uvGsa.res
```

uvSignif *Filter significant terms in the univariate gene set analysis.*

Description

Filters the rows in the data.frame returned by uvGsa so that only the enriched blocks are kept.

Usage

```
uvSignif(gsaout, cutoff = 0.05, pvalue = "padj", statistic = "lor",
  verbose = TRUE, sort = TRUE)
```

Arguments

gsaout	data.frame; output from uvGsa.
cutoff	p-value cutoff for considering significant a Gene Set.
pvalue	p-value column to be used. Default is named "padj" as in uvGsa output.
statistic	name of the column containing the log odds ratio from the uvGsa analysis.
verbose	verbose
sort	if TRUE the output data.frame is ordered according to significance.

Details

Works as goLeaves but without removing redundant terms.

Value

The input data.frame but keeping just the 'significant' functional blocks.

Author(s)

David Montaner <dmontaner@cipf.es>

See Also

uvGsa, uvPat, propagateGO, pval2index, goLeaves

Examples

```
uvGsa.res <- as.data.frame (list (N    = c (10, 20, 30, 40),
                                  lor  = c (1.45, -0.32, 1.89, -1.66),
                                  pval = c (0.001, 0.002, 0.05, 0.06)))
uvGsa.res[,"padj"] <- p.adjust (uvGsa.res$pval, "BY")
uvGsa.res
uvSignif (uvGsa.res)

## Not run:
res <- uvGsa (rindex, annotList)
uvSignif (res)

## End(Not run)
```

Index

∗ **GO**
getGOnames, 5
getOntology, 6
propagateGO, 13
splitOntologies, 16
∗ **GSA**
mdGsa, 9
mdPat, 10
plotMdGsa, 12
uvGsa, 18
uvPat, 20
∗ **KEGG**
getKEGGnames, 5
∗ **annotation**
annotFilter, 2
annotList2mat, 3
annotMat2list, 4
revList, 15
∗ **child**
goLeaves, 7
∗ **enriched**
uvSignif, 21
∗ **filter**
annotFilter, 2
∗ **genes**
transferIndex, 17
∗ **gene**
mdGsa, 9
propagateGO, 13
uvGsa, 18
∗ **go**
goLeaves, 7
∗ **index**
indexTransform, 8
pval2index, 14
transferIndex, 17
∗ **leaves**
goLeaves, 7
∗ **list**
annotFilter, 2
annotList2mat, 3
annotMat2list, 4
revList, 15
∗ **matrix**
annotList2mat, 3
annotMat2list, 4
∗ **miRNAs**
transferIndex, 17
∗ **multidimensional**
mdGsa, 9
mdPat, 10
plotMdGsa, 12
∗ **multivariate**
mdGsa, 9
mdPat, 10
plotMdGsa, 12
∗ **names**
getGOnames, 5
getKEGGnames, 5
getOntology, 6
∗ **ontology**
getGOnames, 5
getOntology, 6
propagateGO, 13
splitOntologies, 16
∗ **p-value**
pval2index, 14
∗ **pattern**
mdPat, 10
uvPat, 20
∗ **plot**
plotMdGsa, 12
∗ **propagate**
propagateGO, 13
∗ **ranking**
indexTransform, 8
pval2index, 14
∗ **revert**
revList, 15
∗ **set**
mdGsa, 9
uvGsa, 18
∗ **significant**
uvSignif, 21
∗ **split**
splitOntologies, 16

∗ **terms**
goLeaves, 7
uvSignif, 21
∗ **to**
transferIndex, 17
∗ **transfer**
transferIndex, 17
∗ **transform**
indexTransform, 8
∗ **univariate**
uvGsa, 18
uvPat, 20

annotFilter, 2, *13*
annotList2mat, 3, *4*
annotMat2list, *3, 4*, 4, *13*, *16*

getGOnames, 5, *6, 7*
getKEGGnames, *5*, 5, *7*
getOntology, *5*, 6
goLeaves, *5*, *7*, 7, *17*, *21*

indexTransform, 8, *15*, *18*

mdGsa, 9, *11*, *13*, *18, 19*
mdPat, *8*, *10*, 10, *13*
multidimensionalGsa *(*mdGsa*)*, 9
multivariateGsa *(*mdGsa*)*, 9

plotMdGsa, 12
propagateGO, *5*, *7*, 13, *17*, *21*
pval2index, *7*, 14, *21*

revList, *4*, 15, *16*

splitOntologies, *5*, *7*, 16

transferIndex, *8*, 17

univariateGsa *(*uvGsa*)*, 18
uvGsa, *7, 8*, *10*, *18*, 18, *20, 21*
uvPat, *7*, *11*, *19*, 20, *21*
uvSignif, 21