Package 'GSNA'

December 7, 2023

Type Package

Title Gene Set Networking Analysis Package (Title Case)

Version 0.1.0

Description This package provides functions for performing Gene Set Networking Analysis (GSNA). It is designed to work in tandem
with standard pathways analysis methods, such as CERNO, GSEA and others, for simplifying such data sets by grouping together
related functions on the basis of shared genes.
Inputs to GSNA are the outputs of pathways analysis methods: a list of gene sets, modules, pathways or GO-terms with associated
p-values. Since these pathways analysis methods may be used to analyze many different types of data including transcriptomic,
metagenomic, and high-
throughput screen data sets, the GSNA pipeline is useful for analysing these data as well.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports circlize,
DT,
dendextend,
dplyr,
ggplot2,
graphics,
grDevices,
igraph,
inline,
Matrix,
methods,
psych,
raster,
stringr,
stringi,
stats,
tibble,
tidyr,

tmod,
utils,
withr,
Rcpp

Depends R (>= 3.0.0)

Suggests gplots,
knitr,
rmarkdown,
testthat (>= 3.0.0)

VignetteBuilder knitr

LinkingTo Rcpp

Config/testthat/edition 3

R topics documented:

adjust_plt *adjust_plt*

Description

adjust_plt

Usage

```
adjust_plt(
  .plt,
  y.dim.actual.fu,
  x.dim.actual.fu,
  v.adjust = "top",
  h.adjust = "center",
  v.strict = TRUE,
  h.strict = TRUE
)
```

Arguments

.plt	The current reserved plot area in the format of the 'plt' argument of par().
y.dim.actual.fu	The required y dimension.
x.dim.actual.fu	The required x dimension.
v.adjust	(optional) a string telling the function how to adjust the legend plotting area. Acceptable values are 'top', 'bottom', and 'middle', indicating that the plot area should be adjusted to be flush with the top or bottom of the available plot area, or otherwise centered, respectively. (default: 'top')
h.adjust	(optional) a string telling the function how to adjust the legend plotting area. Acceptable values are 'left', 'right', and 'center', indicating that the plot area should be adjusted to be flush with the left or right edge of the available plot area, or otherwise centered, respectively. (default: 'center')
v.strict	(optional) Boolean value that if TRUE, tells the function to check that the available virtical space is greater or equal to the y.dim.actual.fu. If FALSE, than the adjusted vertical dimension may end up being greater than allocated space. (default: TRUE)
h.strict	(optional) Boolean value that if TRUE, tells the function to check that the available horizontal space is greater or equal to the x.dim.actual.fu. If FALSE, than the adjusted horizontal dimension may end up being greater than allocated space. (default: TRUE)

Details

This function adjusts the reserved plot areas legends to the proportions required for rendering.

Value

The returned value is a plot area boundaries in the format of the 'par' value returned by the par() function.

adj_mar_leg_vm *adj_mar_leg_vm*

Description

This utility function scales the virtual legend margin for figures width smaller than threshold inches (defaults to 5 inches) or height (defaults to 2.5 inches). To scale, the the function determines whether the width and height fall below their respective thresholds. If the ratios are less than 1, then the margins are scaled to whichever has the smaller ratio.

This is a hack, since we should be scaling, not to the total size of the figure, but to the size available for the legend itself.

Usage

```
adj_mar_leg_vm(
  .mar.leg.vm,
  width = par("fin")[1],
  height = par("fin")[2],
  width.threshold = 5,
  height.threshold = 2.5
)
```

Arguments

.mar.leg.vm	Input virtual legend margins.
width	(optional) Figure width. (defaults to par('fin')[1])
height	(optional) Figure height. (defaults to par('fin')[2])
width.threshold	(optional) The width threshold for scaling. Below this threshold, scaling is performed.
height.threshold	(optional) The height below threshold for scaling. Below this threshold, scaling is performed.

Details

The default behavior of this function is to scale margins to the total size of the figure, but it should actually be scaling the margins to the size of the legend itself.

Value

A new .mar.leg.vm that is scaled for the current plot legend.

Examples

```
## Not run:
.mar.leg.vm <- adj_mar_leg_vm( .mar.leg.vm = c( 4.1, 4.1, 2.1, 2.1 ),
                               width = 1.5,
                               height = 1.5,
                               width.threshold = 1.8,
                               height.threshold = 1.8 )
```

```
## End(Not run)
```

`antiSplit` *antiSplit*

Description

Convert a list of vectors to a data.frame. This method does the opposite of the R base split, but more conveniently than unsplit.

Usage

```
antiSplit(.l, col.names = c("V1", "V2"))
```

Arguments

`.l`	A list with named elements that are vectors.
`col.names`	The names of the output columns. Defaults to col.names = c("V1","V2").

Value

A data.frame is returned with two character columns. The list element names become the first column, whereas the values within the vectors become the second column.

This is used by `assignSubnets()`. We're not currently exporting it.

Examples

```
library(GSNA)
data.l<-list( A = c( 1, 2, 3, 4 ), B = c( 3, 6 ), C = c( 7, 3, 2 ) )
data.df <- GSNA:::antiSplit( data.l, c("Letters", "NumsAsCharacters") )
```

`assignSubnets` *assignSubnets*

Description

Utility function for assigning subnets. Not usually called directly by most users. Instead, use `gsnAssignSubnets()`.

Usage

```
assignSubnets(edges.df, scoreCol = NULL, highToLow = NULL)
```

Arguments

edges.df	A data frame with at least 2 columns. The first two character vector columns indicate vertices. Additional columns are numeric scores for ranking edges.
scoreCol	(optional) A score column used for ordering edges. See explanation below. If there are 3 or more columns the last one is presumed to be the score column and used for ordering. The score is usually derived from a pathways score but may also be derived the pared distance matrix.
highToLow	(optional) A boolean indicating how scores are to be ordered based on significance, low to high, or high to low.

Details

The assignSubnets method uses distances derived from pathways data or from the pared distance matrix to join subnets, starting with the most significant edge scores in a subnet, and subsequently joining additional vertices in order of the best score.

Value

The function returns a list containing:

edges The edges data.frame, but with a subnet column added.

subnets A list of vectors such that the names of the vectors are the names of subnets, and the contents of each vector are the gene sets making up that vector.

vertex_subnets A data.frame containing the name of a vertex and its assigned subnet.

See Also

gsnAssignSubnets

Examples

```
## Not run:
  subnets.l <- assignSubnets( edges.df = edges.df, scoreCol = "p.adj", highToLow = FALSE )

## End(Not run)
```

---

buildGeneSetNetworkGeneric
buildGeneSetNetworkGeneric

Description

General function to create a GSNData object and calculate a distance matrix within. Employed by buildGeneSetNetworkSTLF(), buildGeneSetNetworkLF(), buildGeneSetNetworkJaccard() and buildGeneSetNetworkJaccard() functions.

Usage

```
buildGeneSetNetworkGeneric(
  object = NULL,
  ref.background = NULL,
  geneSetCollection = NULL,
  distMatrixFun,
  distance,
  optimal_extreme
)
```

Arguments

`object`	An object of type GSNData. If NULL, a new one is instantiated.
`ref.background`	(required) A character vector corresponding to the genes observable in a differential expression, ATACSeq or other dataset. This corresponds to the background used in tools like DAVID.
`geneSetCollection`	(required) A gene set collection either in the form of a tmod object, or a list of gene sets / modules as character vectors containing gene symbols and names corresponding to the gene module identifier.
`distMatrixFun`	(required) Function for calculating the distance matrix. Functions used for this purpose are expected to return a square numeric matrix corresponding to the distances between all gene sets. (see `scoreLFMatrix_C`, `scoreJaccardMatrix_C`, `scoreOCMatrix_C`)
`distance`	(required) Name of the distance matrix being calculated.
`optimal_extreme`	(required) Indicates whether max or min values are most significant in the specified distance matrix. Can be `'max'` or `'min'`.

Details

In most cases, users will want to run the specific `buildGeneSetNetworkSTLF()`, `buildGeneSetNetworkLF()`, `buildGeneSetNetworkJaccard()` or `buildGeneSetNetworkJaccard()` functions instead of this, but this function can be used for adding support for new distance metrics.

Value

This function returns a GSNData object.

See Also

`buildGeneSetNetworkJaccard`

`buildGeneSetNetworkOC`

`buildGeneSetNetworkLF`

`buildGeneSetNetworkSTLF`

Examples

```
library(GSNA)

## Not run:
observable_genes <- rownames(FILTERED_RNASEQ_COUNT_MATRIX)
msig.subset <- msig[cerno_results$ID,]
GSN <- buildGeneSetNetworkGeneric( object = NULL,
                                   ref.background = observable_genes,
                                   geneSetCollection = msig.subset,
                                   distMatrixFun = scoreLFMatrix_C,
                                   distance = 'stlf',
                                   optimal_extreme = "min"
                                  )

## End(Not run)
```

`buildGeneSetNetworkJaccard` *buildGeneSetNetworkJaccard*

Description

Using a gene set collection and a background of observable genes, calculate a matrix of Jaccard similarity indices and return a GSNData object.

Usage

```
buildGeneSetNetworkJaccard(
  object = NULL,
  ref.background = NULL,
  geneSetCollection = NULL,
  distMatrixFun = scoreJaccardMatrix_C
)
```

Arguments

`object`	An object of type GSNData. If NULL, a new one is instantiated.
`ref.background`	(required) A character vector corresponding to the genes observable in a differential expression, ATACSeq or other dataset. This corresponds to the background used in tools like DAVID.
`geneSetCollection`	(required) A gene set collection either in the form of a tmod object, or a list of gene sets / modules as character vectors containing gene symbols and names corresponding to the gene module identifier.
`distMatrixFun`	(optional) Function for calculating the distance matrix. Defaults to `scoreJaccardMatrix_C`. Functions used for this purpose are expected to return a square numeric matrix corresponding to the distances between all gene sets.

Details

This function wraps the process of creating a GSNData object and calculating a Jaccard similarity matrix. The Jaccard index matrix is calculated using `scoreJaccardMatrix()`, which is implemented in C++.

Note: Because with Jaccard similarity indices, higher values indicate a closer match between sets, they are unlike standard metrics of distance. Therefore the optimal_extreme is `"max"`, and for certain operations, such as construction of a hierarchical tree, they may require transformation for use in clustering.

Value

This function returns a GSNData object with the `$default_distance` field set as `'jaccard'` and `$distances$lf$optimal_extreme` set to `'max'`.

See Also

`scoreJaccardMatrix_C` `buildGeneSetNetworkLFFast` `buildGeneSetNetworkSTLF`

Examples

```
library(GSNA)

## Not run:
observable_genes <- rownames(FILTERED_RNASEQ_COUNT_MATRIX)
msig.subset <- msig[cerno_results$ID,]
GSN <- buildGeneSetNetworkJaccard( object = NULL,
                                   ref.background = observable_genes,
                                   geneSetCollection = msig.subset )

## End(Not run)
```

---

buildGeneSetNetworkLF *buildGeneSetNetworkLF, buildGeneSetNetworkLFFast-deprecated*

Description

Using a gene set collection and a background of observable genes, calculate log partial Fisher *p*-value distances and return the results as a GSNData object. This is equal to

log(P) = log((a+b)!(c+d)!(a+c)!(b+d)! / (a!b!c!d!(a+b+c+d)!))

This differs from the `buildGeneSetNetworkSTLF` in that only the one value of P is summed, whereas in `buildGeneSetNetworkSTLF`, all more extreme values are summed (prior to log-transformation), generating an actual single-sided *p*-value.

This statistic behaves approximately like a 2-sided Fisher exact test, but may not be appropriate for most purposes. It is also somewhat faster to calculate than STLF (single tailed log-Fisher). Unless speed is an issue, we recommend using `buildGeneSetNetworkSTLF` Note: `buildGeneSetNetworkLFFast` is deprecated. Please use `buildGeneSetNetworkLF()` instead.

Usage

```
buildGeneSetNetworkLF(
  object = NULL,
  ref.background = NULL,
  geneSetCollection = NULL,
  distMatrixFun = function(geneSetCollection) scoreLFMatrix_C(geneSetCollection,
    alternative = 4)
)

buildGeneSetNetworkLFFast(...)
```

Arguments

`object`	An object of type GSNData. If NULL, a new one is instantiated.
`ref.background`	(required) A character vector corresponding to the genes observable in a differential expression, ATACSeq or other dataset. This corresponds to the background used in tools like DAVID. This is **required**, unless object already exists and contains a genePresenceAbsence matrix field.
`geneSetCollection`	(required) A gene set collection either in the form of a tmod object, or a list of gene sets / modules as character vectors containing gene symbols and names corresponding to the gene module identifier. This is **required**, unless object already exists and contains a genePresenceAbsence matrix field.
`distMatrixFun`	Function used to calculate distances. Takes a genePresenceAbsence matrix and returns a distance matrix. (defaults to scoreLFMatrix_C)
`...`	: Arguments passed to `buildGeneSetNetworkLF()`.

Details

This function wraps the process of creating a GSNData object and calculating a log Fisher *p*-value distance matrix. The distance matrix is calculated using `scoreLFMatrix_C()`.

Value

This function returns a GSNData object with the `$default_distance` field set as `'lf'` and `$distances$lf$optimal_ex` set to `'min'`.

Functions

- `buildGeneSetNetworkLFFast()`: Deprecated, use `buildGeneSetNetworkLF()`.

`buildGeneSetNetworkLFFast`

For `buildGeneSetNetworkLFFast()`, use `buildGeneSetNetworkLF()`.

See Also

`scoreLFMatrix_C` `scoreJaccardMatrix_C` `scoreOCMatrix_C`

Examples

```
## Not run:
library(GSNA)
observable_genes <- rownames(FILTERED_RNASEQ_COUNT_MATRIX)
msig.subset <- msig[cerno_results$ID,]
GSN <- buildGeneSetNetworkLF( object = NULL,
                              ref.background = observable_genes,
                              geneSetCollection = msig.subset )

## End(Not run)
```

`buildGeneSetNetworkOC` *buildGeneSetNetworkOC*

Description

Using a gene set collection and a background of observable genes, calculate a matrix of Szymkiewicz–Simpson overlap coefficients and return a GSNData object.

Usage

```
buildGeneSetNetworkOC(
  object = NULL,
  ref.background = NULL,
  geneSetCollection = NULL,
  distMatrixFun = scoreOCMatrix_C
)
```

Arguments

`object`	An object of type GSNData. If NULL, a new one is instantiated.
`ref.background`	(required) A character vector corresponding to the genes observable in a differential expression, ATACSeq or other dataset. This corresponds to the background used in tools like DAVID.
`geneSetCollection`	(required) A gene set collection either in the form of a tmod object, or a list of gene sets / modules as character vectors containing gene symbols and names corresponding to the gene module identifier.
`distMatrixFun`	(optional) Function for calculating the distance matrix. Defaults to `scoreOCMatrix_C`. Functions used for this purpose are expected to return a square numeric matrix corresponding to the distances between all gene sets.

Details

This function wraps the process of creating a GSNData object and calculating a Szymkiewicz–Simpson overlap coefficient matrix. The Szymkiewicz–Simpson overlap coefficient matrix is calculated using `scoreOCMatrix()`, which is implemented in C++.

Note: Because with Szymkiewicz–Simpson overlap coefficients, higher values indicate a closer match between sets, they are unlike standard metrics of distance. Therefore the optimal_extreme is `"max"`, and for certain operations, such as construction of a hierarchical tree, they may require

transformation for use in clustering. Secondly, since the Szymkiewicz–Simpson method often produces a large number of tie values in a distance matrix, we recommend paring using hierarchical clustering (with gsnPareNetGenericHierarchic) instead of nearest neighbor clustering.

Value

This function returns a GSNData object with the $default_distance field set as 'oc' and $distances$lf$optimal_ex set to 'max'.

See Also

scoreOCMatrix_C buildGeneSetNetworkJaccard buildGeneSetNetworkLFFast buildGeneSetNetworkSTLF

Examples

```
library(GSNA)

## Not run:
observable_genes <- rownames(FILTERED_RNASEQ_COUNT_MATRIX)
msig.subset <- msig[cerno_results$ID,]
GSN <- buildGeneSetNetworkOC( object = NULL,
                              ref.background = observable_genes,
                              geneSetCollection = msig.subset )

## End(Not run)
```

---

buildGeneSetNetworkSTLF *buildGeneSetNetworkSTLF*

Description

Using a gene set collection and a background of observable genes, calculate single-tailed log Fisher *p*-value distances and return the results as a GSNData object.

Usage

```
buildGeneSetNetworkSTLF(
  object = NULL,
  ref.background = NULL,
  geneSetCollection = NULL,
  distMatrixFun = scoreLFMatrix_C
)
```

Arguments

object	An object of type GSNData. If NULL, a new one is instantiated.
ref.background	A character vector corresponding to the genes observable in a differential expression, ATACSeq or other dataset. This corresponds to the background used in tools like DAVID. This is **required**, unless object already exists and contains a genePresenceAbsence matrix field.

`geneSetCollection`
: (required) A gene set collection either in the form of a tmod object, or a list of gene sets / modules as character vectors containing gene symbols and names corresponding to the gene module identifier. This is **required**, unless object already exists and contains a genePresenceAbsence matrix field.

`distMatrixFun`
: Function used to calculate distances. Takes a genePresenceAbsence matrix and returns a distance matrix. (defaults to scoreLFMatrix_C)

Details

This function wraps the process of creating a GSNData object and calculating a log Fisher *p*-value distance matrix. The distance matrix is calculated using `scoreLFMatrix_C ()`, which is currently implemented in R and about eight- to tenfold slower than `buildGeneSetNetworkLFFast()`.

Value

This function returns a GSNData object with the `$default_distance` field set as `'stlf'` and `$distances$lf$optimal_extreme` set to `'min'`.

See Also

`scoreLFMatrix_C` `buildGeneSetNetworkLFFast`, `buildGeneSetNetworkJaccard`

Examples

```
## Not run:
library(GSNA)

observable_genes <- rownames(FILTERED_RNASEQ_COUNT_MATRIX)
msig.subset <- msig[cerno_results$ID,]
GSN <- buildGeneSetNetworkSTLF( object = NULL,
                                ref.background = observable_genes,
                                geneSetCollection = msig.subset )

## End(Not run)
```

`color2IntV` *color2IntV*

Description

Convert a color, either as a name or as a RGB hexedecimal value to an integer vector containing the RGB specification.

Usage

```
color2IntV(color)
```

Arguments

`color`
: A color specified either by name (e.g. "red") or as a RGB hexadecimal value (e.g. "#FF0000").

Value

A integer vector containing the RGB specification.

See Also

intV2Color()

---

combineRGBMatrices *combineRGBMatrices*

Description

Given 2 different matrices of colors, combine the colors numerically. This is used in makeTwoColorEncodeFunction().

Usage

```
combineRGBMatrices(
  c1.mat,
  c2.mat,
  combine_method = "scaled_geomean",
  max_per_channel = 255
)
```

Arguments

c1.mat	A numeric matrix with three columns corresponding to red, green, blue, with numerical values ranging from 0 to 255 to be combined numerically with c2.mat.
c2.mat	A numeric matrix with three columns corresponding to red, green, blue, with numerical values ranging from 0 to 255 to be combined numerically with c1.mat.
combine_method	Method of combining colors, can be 'scaled_geomean', 'standard'/'euclidean', 'negative_euclidean', 'mean', 'scaled_geomean' (default = 'scaled_geomean')
max_per_channel	Maximal color value per RGB channel (default 255).

Value

A 3-column RGB matrix of combined colors.

See Also

makeTwoColorEncodeFunction()

Examples

```
c1.mat <- matrix( c(255, 100, 0 ), ncol = 3 )
c2.mat <- matrix( c( 0, 50, 255 ), ncol = 3 )
c12.mat <- combineRGBMatrices( c1.mat, c2.mat, "euclidean" )
```

contrasting_color *contrasting_color*

Description

Function picks colors to contrast with the colors given as the 'col' argument.

Usage

```
contrasting_color(
  col,
  type = "complement",
  threshold = 127,
  low = "#000000",
  high = "#FFFFFF"
)
```

Arguments

col	A character vector of colors.
type	(optional) Type of contrasting color, i.e. the method of generating the contrasting color. Valid values are 'complement', 'rotate', 'yellow', 'gray', 'binary' and 'blackyellow'.
threshold	(optional, used only for type='binary') The "binary" method works by assessing the mean value of the RGB channels. If the value is above a threshold, the low color is returned, if it is below the threshold, the high color is returned.
low	(optional, used only for type='binary') Low color (see threshold argument).
high	(optional, used only for type='binary') High color (see threshold argument).

Value

A contrasting color.

distMat2Rank *distMat2Rank*

Description

Convert a symmetrical numerical matrix of distances to a matrix of ranks. Note: Only the lower side of the matrix is used. Data on the upper right are assumed to be redundant.

Usage

```
distMat2Rank(mat, lower_is_closer = TRUE)
```

Arguments

mat	A numerical matrix containing distances.
lower_is_closer	Logical indicating that lower is closer.

Details

This is used by default by gsnPareNetGenericHierarchic.

Value

A symmetric matrix of scaled ranks (quantiles). The matrix has the attribute 'lower_is_closer' set to 'TRUE'.

See Also

`distMat2UnitNormRank()`

Examples

```
## Not run:
mat.dist <-  matrix( c( NA, -400, -600, NA, NA, -120, NA, NA, NA ), nrow = 3, ncol = 3 )
mat.ranks <- distMat2Rank(mat.dist)

## End(Not run)
```

---

distMat2UnitNormRank *distMat2UnitNormRank negDistMat2UnitNormRank*

Description

Convert a symmetrical numerical matrix of distances to a matrix of scaled ranks (from 0 to 1). Note: Only the lower side of the matrix is used. Data on the upper right are assumed to be redundant. These functions are intended to convert a matrix of distance or similarity values into a proper form for applying hierachical clustering with the `gsnPareNetGenericHierarchic()` function.

Usage

```
distMat2UnitNormRank(mat, lower_is_closer = TRUE)

negDistMat2UnitNormRank(mat)
```

Arguments

`mat`	A numerical matrix containing distances.
`lower_is_closer`	Logical indicating that lower is closer.

Details

The difference between `distMat2UnitNormRank()` and `negDistMat2UnitNormRank()` is that `negDistMat2UnitNormRa` takes only the `mat` argument, and negates it prior to calling `distMat2UnitNormRank()`.

Value

A symmetric matrix of ranks. The matrix has the attribute 'lower_is_closer' set to 'TRUE'.

Functions

- `negDistMat2UnitNormRank()`: Takes the same parameter distMat2UnitNormRank, but multiplies the distance by -1 first.

See Also

`distMat2Rank()`

Examples

```
## Not run:
mat.dist <- matrix( c( NA, -400, -600, NA, NA, -120, NA, NA, NA ), nrow = 3, ncol = 3 )
mat.scaledranks <- distMat2UnitNormRank(mat.dist)

mat.jaccard <- matrix( c( NA, 0.2, 0.3, NA, NA, 0.1, NA, NA, NA ), nrow = 3, ncol = 3 )
mat.srjaccard <- negDistMat2UnitNormRank(mat.jaccard)

## End(Not run)
```

`extract_david_GSC` *extract_david_GSC*

Description

extract_david_GSC

Usage

```
extract_david_GSC(
  data,
  genes.field = "Genes",
  term.field = "Term",
  del = "\\s*,\\s*"
)
```

Arguments

`data`	A data.frame containing DAVID pathways data (without duplicates).
`genes.field`	The name of the field containing the lists of genes for each gene set (default: `"Genes"`).
`term.field`	The name of the field containing the ID for each gene set (default: `"Term"`).
`del`	A pattern specifying the delimiter for the genes in `genes.field`. (default: `"\\s*,\\s*"`)

Value

A gene set collection as a list of gene set vectors, where the gene set names correspond to Terms and the vectors contain gene symbols corresponding to the genes listed in `genes.field`.

See Also

- gsnAddPathwaysData()
- read_david_data_file()

Examples

```
## Not run:
   david.GSC <- extract_david_GSC( data = data.david )

## End(Not run)
```

---

get_usr_x_coords_per_inch *get_usr_x_coords_per_inch*

Description

Internal GSNA package function to return the ratio of user x coordinates per inch. This number is used in the makeNodeSizeLegend() functions.

Usage

```
get_usr_x_coords_per_inch()
```

Value

The numerical value corresponding to par('usr')[2] - par('usr')[1]) / par('pin')[1]

Examples

```
## Not run:
    uxcpi <- get_usr_x_coords_per_inch()

## End(Not run)
```

---

gsc2tmod *gsc2tmod*

Description

Function to convert a GSC in the form of a named list of vectors containing gene symbols to a object of class tmod which was used by the tmod prior to version 0.50.11,

Usage

```
gsc2tmod(MODULES2GENES, MODULES = NULL, GENES = NULL)
```

Arguments

MODULES2GENES	A named list of character vectors in which the vectors correspond to gene sets and contain gene symbols (or other gene identifiers) and the names are the corresponding gene set identifiers.
MODULES	(optional) A data.frame containing an ID and a Title field in the same order as the gene sets in MODULES2GENES. Furthermore, the row names should (apparently) correspond to the IDs in the corresponding rows. If not provided, this will be generated automatically.
GENES	(optional) A data frame with gene metadata. Must contain an ID column. If not provided, this will be generated automatically.

Value

Returns a tmod object.

See Also

read_gmt() tmod2gsc()

Examples

```
## Not run:
  gsc <- read_gmt( "gene_set_collection.GMT" )
  gsc.tmod <- gsc2tmod( gsc )

## End(Not run)
```

---

gsIntersect *gsIntersect*

Description

For two character vectors, returns the set of shared elements. This is used by GSNA to find shared genes in two gene sets.

Usage

```
gsIntersect(gs1, gs2)
```

Arguments

gs1	A character vector representing gene symbols in a gene set.
gs2	A character vector representing gene symbols in a second gene set.

Details

This version of the function is used in gsnORAtest_cpp. (In another version of the function, used in gsnFilterGeneSetCollectionList() and accessible only from C++ the first argument is gs1Set, a set of strings of type std::set<std::string>.)

This function does essentially what R's base::intersect does, so it is not necessarily useful to export.

Value

A character vector consisting of the overlap of the two gene sets.

Examples

```
## Not run:
  gs12.sharedgenes <- gsIntersect( gs1, gs2 )

## End(Not run)
```

`gsIntersectCounts` *gsIntersectCounts*

Description

For two character vectors representing two gene sets (gs1 and gs2) and a total number of background observable genes (that may also be present in gs1 and or gs2 or neither), this function calculates the counts in a 2x2 contingency table for presence and absence of genes in one or both sets or neither. The output of this function is used as the input for a Fisher test calculation by the GSNA package.

Usage

```
gsIntersectCounts(gs1, gs2, bg_size)
```

Arguments

`gs1`	A character vector representing gene symbols in a gene set.
`gs2`	A character vector representing gene symbols in a second gene set.
`bg_size`	An integer representing the size of the background, i.e. the total number of observable genes.

Details

This version of the function may not be retained since it's not currently used. Two alternative versions of the function in C++ that find the overlap between a `std::set<std::string>` and a character vector are used since those versions are much faster.

NOTE: This function assumes that all genes in gs1 and gs2 are present in the background, so to use this properly, gs1 and gs2 must be filtered to include only genes present in the background.

Value

A numeric vector of length 4 containing the following 4 elements:

1	The number of genes in the background that are absent in gs1 and gs2.
2	The number of genes in gs1 but not gs2.
3	The number of genes in gs2 but not gs1.
4	The number of genes in in both gs1 and gs2.

Examples

```
## Not run:
  gs12.sharedgenecount <- gsIntersectCounts( gs1, gs2 )

## End(Not run)
```

`gsnAddPathwaysData` *gsnAddPathwaysData*

Description

Add pathways search data to a GSNData object.

A synonym of `gsnAddPathwaysData()`, included to support old code. Use `gsnAddPathwaysData()` for new code.

Usage

```
gsnAddPathwaysData(
  object,
  pathways_data,
  type = NULL,
  id_col = NULL,
  stat_col = NULL,
  sig_order = NULL,
  stat_col_2 = NULL,
  sig_order_2 = NULL,
  n_col = NULL
)

gsnAddPathwayData(...)
```

Arguments

`object`	A GSNData object.
`pathways_data`	A data.frame containing the results of pathways analysis.
`type`	(optional) A character vector of length 1 indicating the type of pathways data being added to the GSNData object. This can be `'cerno'`, `'gsea'`, `'gsnora'`, or other arbitrary types. If not explicitly indicated, the method attempts to examine the column names of the data.frame in order to determine what kind of import to perform, then calls other methods for the actual import. For `'cerno'`, `'gsea'`, and `'gsnora'`, the actual import is performed by methods specifically designed for CERNO and GSEA import. Otherwise a method for generic import is used.
`id_col`	(optional) A character vector of length 1 indicating the name of the column used as a key for gene sets/modules. This corresponds to the ID field of `tmod` objects, or the names of vectors in a list vectors gene sets/modules, both of which can be used as a geneSetCollection argument in building gene set networks. In the case of CERNO and GSEA data sets, there are preset values for `id_col`, but in the case of generic import, the import method attempts to guess. If an ID cannot be inferred, then an error is thrown.

stat_col	(optional) A character vector of length 1 indicating the name of the column used as a statistic to evaluate the quality of pathways results. This is generally a *p*-value of some sort. In the case of CERNO and GSEA data sets, there are preset values for stat, but in the case of generic import, the import method attempts to guess.
sig_order	(optional) Either 'loToHi' or 'hiToLo' depending on the statistic used to evaluate pathways results. For *p*-values, this should be 'loToHi'.
stat_col_2	(optional) A character vector of length 1 indicating the name of the column used as a second statistic to evaluate pathway result quality. Used in 2-color networks.
sig_order_2	(optional) Either 'loToHi' or 'hiToLo' depending on stat_col_2. Used in 2-color networks.
n_col	(optional) Specifies the column containing the number of genes in the gene set. Generally, this is the number of genes in the gene set that are attested in an expression data set.
...	Arguments passed on to gsnAddPathwaysData

Details

Pathways data are used by the assignSubnets() function, which organizes subnets on the basis of this statistic. If sig_order is 'loToHi', and the evaluation statistic ('stat') is a *p*-value, then the first node in each subnet will be the node with the lowest *p*-value, for example. This ordering is not an absolute requirement.

This is provided to simplify workflows and facilitate imports that can identify and handle multiple types of pathways data, but also the CERNO, GSEA, GSNORA, and generic import methods can be used directly (gsnImportCERNO, gsnImportGSEA, gsnImportGSNORA, and gsnImportGenericPathways).

Notes: These import handlers perform checks on the provided pathways data to verify that all gene set IDs in the genePresenceAbsence matrix are present in the ID column of the pathways data. An error is thrown if all gene set IDs in the genePresenceAbsense are not present in the pathways ID column. On the other hand, if there are gene set IDs present in the pathways data that are absent from the genePresenceAbsence matrix, then thes methods emit a warning.

Value

This returns a GSNData object containing imported pathways data.

Functions

- gsnAddPathwayData(): Synonym of gsnAddPathwaysData(), included to support old code. Use gsnAddPathwaysData() for new code.

See Also

gsnImportCERNO gsnImportGSEA gsnImportGSNORA gsnImportGenericPathways

Examples

```
## Not run:
gsn_object <- gsnAddPathwaysData( object = gsn_object, pathways_data = dat.cerno )

## End(Not run)
```

gsnAssignSubnets *gsnAssignSubnets*

Description

Main wrapper method for assigning subnets.

Usage

```
gsnAssignSubnets(object, distance = NULL, scoreCol = NULL, highToLow = NULL)
```

Arguments

object	An object of type GSNData containing pathways data and a pared distance matrix.
distance	(optional) character vector of length 1 indicating which pared distance matrix is to be used for assigning subnets. This defaults to the 'default_distance'.
scoreCol	(optional) A score column used for ordering edges. See explanation below. If there are 3 or more columns the last one is presumed to be the score column and used for ordering. The score is usually derived from a pathways score but may also be derived the pared distance matrix.
highToLow	(optional) A boolean indicating how scores are to be ordered based on significance, low to high, or high to low.

Details

Calls assignSubnets method using scores derived from pathways data, starting with the most significant edge scores in a subnet, and subsequently joining additional vertices in order of the best score.

Value

The method returns a GSNData object containing the following data for the indicated distance matrix:

edges The edges data.frame, but with a subnet column added.

subnets A list of vectors such that the names of the vectors are the names of subnets, and the contents of each vector are the gene sets making up that vector.

vertex_subnets A data.frame containing the name of a vertex and its assigned subnet.

See Also

assignSubnets

Examples

```
## Not run:
  subnets.l <- assignSubnets( edges.df = edges.df, scoreCol = "p.adj", highToLow = FALSE )

## End(Not run)
```

GSNData *GSNData*

Description

GSNData object constructor, used to generate new GSNData objects.

Usage

```
GSNData(distances = list(), ...)
```

Arguments

distances	Optional parameter containing a list of module-module distance metric data organized by the name of the distance metric used, e.g. lf, jaccard, stlf.
...	Additional arguments. Object fields can be set as named arguments this way using name = value pairs.

Details

This method is called by `buildGeneSetNetworkLFFFast()`, `buildGeneSetNetworkLFFast()` and `buildGeneSetNetworkSTLF()`. For most users there will be little reason to call this method except when tying to implement support for new distance metrics or utility functions.

Value

A new GSNAData object.

Structure of the GSNData object:

The GSNData object can contain multiple distance matrices including log Fisher (lf) and Jaccard (jaccard). These distances, along with associated pared-distances, and significance order parameters are stored in named sublists within the `$distances` lists, the sublists are named after their respective distance metric (lf, jaccard, etc.) as `$distances[[DIST]]`. These sublists contain a distance matrix `$distances[[DIST]]$matrix`, a significance order `$distances[[DIST]]$optimal_extreme` (e.g. "loToHi" for lf, and "hiToLo" for jaccard), and after paring a `$distances[[DIST]]$pared`.

Fields:

`$GSNA_version` A character vector of length 1 indicqting the version of GSNA used to generate this GSNData object.

`$genePresenceAbsence` A sparse logical Matrix containing presence(TRUE) or absense (FALSE) calls for genes (rows) in gene sets (columns).

`$distances` a named list(). Names indicate a distance metric 'lf', 'jaccard', etc. indicated as `DIST` below.

`$distances[[DIST]]$matrix` A matrix of raw distances

`$distances[[DIST]]$optimal_extreme` Significance order where "min" indicates that low values are optimal/ closer than high values as with log Fisher (lf), and "max" indicates that high values are closer, as with Jaccard (jaccard) distance.

`$distances[[DIST]]$pared_optimal_extreme` Significance order for the pared, scaled distance matrix. This may differ from `$distances[[DIST]]$optimal_extreme` if scaling flips high distance values to low ones, as may be necessary for handling distance matrices such as the Jaccard for which higher values are closer. (See `$distances[[DIST]]$optimal_extreme`, above.)

`$distances[[DIST]]$pared` A pared distance matrix.

`$distances[[DIST]]$edges` A data.frame containing a gathered set of network edges derived from `$distances[[DIST]]$pared`

`$distances[[DIST]]$vertices` A complete list of gene set IDs in the network.

`$default_distance` The default distance used for network construction.

`$ordered_genes` A character vector containing the ordered list of genes in the data set (most important first). This list is also used as the background of observable genes for creating the filteredGeneSetCollection.

`$filteredGeneSetCollection` A filtered set of gene lists (a list of character vectors) containing only the genes present in the differential expression data set. This is the 'background' of all genes observable in the differential expression data.

`$pathways` A named list containing pathways results data, as follows:

`$pathways$data` A data.frame containg a pathways results set.

`$pathways$type` A character vector of length=1 indicating the type of pahways analysis performed, e.g. CERNO, GSEA, ORA.

`$pathways$id_col` Indicates the name of the column in $pathways$data that contains the gene set ID.

`$pathways$stat_col` A character vector of length 1 indicating the statistic used for assessing significance, generally a p-value.

`$pathways$stat_col_2` A character vector of length 1 indicating the statistic used for assessing significance, generally a p-value.

`$pathways$sig_order` Indicates whether low of high values of $pathways$statistic are most significant with "loToHi" indicating that low values are optimal/most significant (as with typical p-values) and "hiToLo" indicating high values are optimal/most significant.

`$pathways$sig_order_2` Indicates whether low of high values of $pathways$statistic are most significant with "loToHi" indicating that low values are optimal/most significant (as with typical p-values) and "hiToLo" indicating high values are optimal/most significant.

`$pathways$n_col` Indicates the name of the pathways column used to indicate effective gene set size, based on genes actually observable in an experimental data set.

Examples

```
library(GSNA)
gsn_obj <- GSNData()
```

gsnDendroSubnetColors *gsnDendroSubnetColors*

Description

Given a list of vectors of gene set IDs corresponding to subnets, returns a vector of colors. with each color corresponding to a subnet (see details).

Usage

```
gsnDendroSubnetColors(subnets)

gsnDendroSubnetColors_dark(subnets)
```

Arguments

subnets	A list of vectors containing, as elements, vectors corresponding to subnets and containing gene set IDs as subnet members. List element names are the names of the subnets. This corresponds to the set of subnets stored in the `$distances[[distance]]$subnets` field of a pared `GSNData` object.

Details

Given a list of vectors in which each vector contains a set of gene set IDs corresponding to a subnet, with list element names being the subnet names, this function generates a vector of colors in which subnets with a single member are colored black and subnets with multiple members are given associated distinct colors. In the returned vector of names, the names are the subnets and the elements are the associated colors. This function is primarily for generating colors for hierarchical dendrograms.

The `gsnDendroSubnetColors()` and `gsnDendroSubnetColors_dark()` do approximately the same thing, but `gsnDendroSubnetColors_dark()` returns a darker palette of colors.

Value

A vector of colors with names corresponding to subnet names.

Examples

```
## Not run:
  analysis.subnets <- analysis.GSN$distances$jaccard$subnets
  colors_v <- gsnDendroSubnetColors( analysis.subnets )

## End(Not run)
```

gsnDistanceHistogram *gsnDistanceHistogram*

Description

Generate a Histogram of Distances

Usage

```
gsnDistanceHistogram(
  object,
  distance = NULL,
  dist.matrix = c("raw", "pared", "edges"),
  stat = "percent",
  colors = NULL,
  bins = 100
)
```

Arguments

object	A GSNData object
distance	A distance, or even a character vector of distances, e.g. c("lf", "jaccard", "stlf").
dist.matrix	The names of distance matrices, which can be "raw" for the distance stored in "matrix", "pared" for the distances stored in "pared", and "edges" for the distances stored in "edges". In general, the distances in "edges" will be the same as those in "pared", but this may not always be true.
stat	Can be "percent", "count", "density", or "cumulative". This determines how the data are visualized.
colors	Currently, this does nothing, but will eventually allow the user to specify custom colors. Stay tuned.
bins	The number of bins, for histograms ("percent" or "count").

Details

This function is useful for such purposes as assessing the effects of paring on the distribution of distances.

Value

A ggplot2 graphical object.

See Also

plot.GSNData gsnPlotNetwork gsnHierarchicalDendrogram

Examples

```
## Not run:
gsnDistanceHistogram( object = GSN,
                      distance = c("pared", "raw" ),
                      stat = "percent",
                      bind = 100 )

## End(Not run)
```

`gsnFilterGeneSetCollectionList` *gsnFilterGeneSetCollectionList*

Description

Given a vector of gene symbols and a gene set collection, filter the gene set collection to include only gene symbols present in the background.

Usage

```
gsnFilterGeneSetCollectionList(bg, geneSetCollection)
```

Arguments

`bg`	A character vector representing gene symbols in a background of observable genes.
`geneSetCollection`	A list of gene sets, in which the gene sets are character vectors containing gene symbols, and the list names are the corresponding gene set identifiers. NOTE: This must be a list, not a `tmod` object. It is trivial to extract such a list from a `tmod` object, however. The `$MODULES2GENES` field of the `tmod` object contains a suitable list.

Details

This function is used in gsnORAtest_cpp to automatically filter the gene set provided. It may be used manually during GSNA analysis.

Value

A filtered gene set as a list of vectors of gene symbols in which the list names correspond to gene set IDs.

Examples

```
## Not run:
  bg <- DE_GENES.df$Gene
  msig_subset_l <- msig$MODULES2GENES[gene_set_ids]
  msig_subset_filt_l <- gsnFilterGeneSetCollectionList( bg, msig_subset_l )

## End(Not run)
```

gsnHierarchicalDendrogram
gsnHierarchicalDendrogram

Description

Generate a dendrogram plot of a hierarchical clustered set of GSNA distances. This requires an embedded hierarchical cluster object of type `'hclust'` associated with the default or specified distance metric. Such an object may be generated by running `gsnPareNetGenericHierarchic()` on a `GSNData` object prior to running this function.

The graphical output of this function can be a horizontal or circular dendrogram. When `show.leaves`, `stat_col` and optionally `stat_col_2`, the function will output a dendrogram image with leaves colored by the significance indicated in `stat_col` and optionally `stat_col_2` (with a 1 or 2 dimensional color scale). If `n_col` is specified, the leaf sizes will be scaled by the column indicated therein.

The function has many optional arguments, but only a few should be necessary to get a decent plot.

Usage

```
gsnHierarchicalDendrogram(
  object,
  distance = NULL,
  subnet_colors = NULL,
  filename = NULL,
  file = NULL,
  out_format = NULL,
  width = NULL,
  height = NULL,
  .mai.plot = NULL,
  cex = par("cex"),
  subnetColorsFunction = gsnDendroSubnetColors_dark,
  id_col = NULL,
  id_nchar = NULL,
  pathways_title_col = c("Title", "Name", "NAME", "STANDARD_NAME"),
  substitute_id_col = NULL,
  font_face = NULL,
  color_labels_by = "subnet",
  show.leaves = FALSE,
  show.legend = TRUE,
  pathways_dat = NULL,
  stat_col = NULL,
  stat_col_2 = NULL,
  sig_order = NULL,
  sig_order_2 = NULL,
  n_col = NULL,
  transform_function = nzLog10,
  leaf_colors = c("white", "yellow", "red"),
  leaf_colors.1 = c("#FFFFFF", "red"),
  leaf_colors.2 = c("#FFFFFF", "blue"),
  leaf_border_color = "#666666",
```

```
  legend.leaf.col = "#CCCCCC",
  combine_method = "scaled_geomean",
  use_leaf_border = TRUE,
  render.plot = TRUE,
  c1.fun = NULL,
  c2.fun = NULL,
  geometry = "horizontal",
  .plt.plot = NULL,
  leaves_pch = NULL,
  leaf_char_shift = 1,
  na.color = "#CCCCCC",
  leaf_cex = NULL,
  leaf_cex_range = c(0.5, 2.1),
  lab.cex = NULL,
  tree_x_size.in = 2,
  legend_x_size.in = 2,
  left_margin.in = 0,
  right_margin.in = NULL,
  top_margin.in = NULL,
  bottom_margin.in = 0,
  legend.downshift.in = NULL,
  bkt_lmargin_chars = 4,
  legend_spacing.x.in = 2 * par("cin")[1],
  legend_spacing.y.in = par("cin")[2],
  legend.lab.cex = NULL,
  legend.axis.cex = NULL,
  legend.free.cex.bool = FALSE,
  main = NULL,
  cex.main = NULL,
  mar.main = 3.2,
  lines.main = 1.5,
  colors.n = 100,
  legend.bg = par("bg"),
  legend.fg = par("fg"),
  draw.legend.box.bool = TRUE,
  DO_BROWSER = FALSE
)
```

Arguments

`object`	An object of the class `GSNData`
`distance`	(optional) A character vector of length one to indicate the desired distance metric to be used for generating a hierarchical dendrogram, e.g. 'lf', 'jaccard', 'stlf', etc. Defaults to the value of objects `default_distance`.
`subnet_colors`	(optional) A character vector of color codes matching the desired colors for subnets. If null then the colors are set automatically.
`filename`	(optional) A file for outputting a graphical image to a file as opposed to the current graphical device. Output format is automatically detected from the file suffix, but can be overridden using the out_format argument. (See details.)
`file`	(optional) Synonym of filename, but deprecated. (Generates a warning.)
`out_format`	(optional) File format of the output, either `'svg'`, `'png'`, `'pdf'`, or `'plot'` (default if filename is not specified). For more information, see Details.

`width` (optional) Used to specify the width of the output in inches. If not specified, defaults to the current figure width.

`height` (optional) Used to specify the height of the output in inches. If not specified, defaults to the current figure height.

`.mai.plot` (optional) A parameter specifying the margins of the plot, excluding legends as inches. This is calculated automatically and for most purposes, will not need to be specified.

`cex` (optional) Font size in cex units. This parameter is used as a basis for setting the various other font sizes including those of leaf/node labels, cluster/subnet labels, and legend text sizes.

`subnetColorsFunction`
(optional) Function for assigning colors to subnets. Only used when `color_labels_by == 'subnet'`. The default value is `gsnDendroSubnetColors_dark`.

`id_col` (optional) Character vector of length 1 indicating the name of the column to be used as an ID key in the pathways dataframe (or modules data if that is used, see below). This column should contain the same values as the names of the gene sets. This defaults to the value of the pathways id_col field.

`id_nchar` (optional) Integer indicating the number of characters to reserve in the dendrogram plot for the ID. If unspecified, it is equal to the maximal nchar of the specified ID (`id_col` or `substitute_id_col`).

`pathways_title_col`
(optional) Character vector of length 1 indicating the name of the column in the pathways or modules data.frame to be used as a Title or descriptor in the plot. If not set the function looks for the following names: "Title", "Name", "NAME", "STANDARD_NAME", and takes the first that it finds. If set to NA, the title part of the label is suppressed.

`substitute_id_col`
(optional) Character vector of length 1 indicating a column used to substitute an alternative ID for the labeling gene sets in data set. If set to `NA`, the ID in the plot is disabled.

`font_face` (optional) The font used for plot text, including leaf labels. For best results, this should be a monospaced font. If not pecified, the system attempts to pick a suitable default: `'Andale Mono'` on Mac OS X, `'Lucida Sans Typewriter'` for Windows, and `'mono'` for all other systems.

`color_labels_by`
(optional) This parameter tells the plotting function to assign colors to dendrogram leaf labels on on the basis of this argument. Currently, only 'subnets' and NULL are supported arguments.

`show.leaves` (optional) Logical to tell the function to display leaves representing gene sets. When `stat_col` and optionally `stat_col_2` are specified, naming parameters from the `pathways_dat` data.table, a single or two-color color scale is used to represent the value of the corresponding pathways statistics.

`show.legend` (optional) A logical value telling the plotting function to include legends.(default: TRUE)

`pathways_dat` (optional) data.frame containing associated pathways data. This defaults to whatever pathways data has already been imported into this GSNData object in `object$pathways$data`.

`stat_col` (optional) This is the name of the column in the pathways data.frame that contains a significance value for coloring network vertices. The default value is specified by `object$pathways$stat_col`.

`stat_col_2` (optional) This is the name of an optional second column in the pathways data.frame that contains a significance value for coloring network vertices in a 2-color network. The default value is specified by `object$pathways$stat_col_2`. When specified, a 2-color network is generated. To force a 2-color network to plot as a standard 1-color network using `stat_col` alone, use `stat_col_2 = NA`.

`sig_order` (optional) This indicates the behavior of `stat_col`, whether low values (`'loToHi'`) or high values (`'hiToLo'`) are most significant. The default value is specified in `object$pathways$sig_order`.

`sig_order_2` (optional) This indicates the behavior of `stat_col`, whether low values (`'loToHi'`) or high values (`'hiToLo'`) are most significant. The default value is specified in `object$pathways$sig_order`.

`n_col` (optional) This is the name of the column in the pathways data.frame that contains a value for gene set size, or any other value intended to be the bases of leaf scaling. When specified, leaf sizes will be scaled by this value. (default is the value in `object$pathways$n_col`). An `NA` value can be used to override the the value in `object$pathways$n_col` and suppress leaf scaling.

`transform_function` (optional) Function to transform significance values for conversion to a color scale. Normally, significance values are *p*-values, and need log transformation. If there are significance values of 0, these are converted to `-Inf` by log-transformation, so the function `nzLog10()` adds a small pseudocount to the values to mitigate this problem, prior to log10 transformation, but for other types of data, other transformations or even 'identity' may be more suitable. (default, `nzLog10`)

`leaf_colors` (optional) A vector containing at least 2 colors for generating a color gradient in single channel visualizations. (default: c("white","yellow","red"), see details)

`leaf_colors.1` (optional) A vector containing at least 2 colors for generating a color gradient in dual channel visualizations. (default: c("white", "red"), see details)

`leaf_colors.2` (optional) A vector containing at least 2 colors for generating a color gradient in dual channel visualizations. (default: c("white", "blue"), see details)

`leaf_border_color` (optional) For R's open plot symbols `pch ( 21, 22, 23, 24, 25 )`, supporting fill with a 'bg' color, leaf border may be specified with this option. (default: "#666666")

`legend.leaf.col` (optional) Leaf fill color for the legend. (default: "#CCCCCC")

`combine_method` (optional) For dual channel plots this is a string used to indicate how colors are combined to generate a two dimensional color scale. Options are "scaled_geomean" (same as "default"), "standard" (same as "euclidean"), "negative_euclidean", "mean", and "additive". See details.

`use_leaf_border` (optional) When automatically choosing a leaf symbol (leaves_pch), this option determines whether a solid or an open symbol is used (see details).

`render.plot` (option) Logical value indicating whether to actually render the plot, or simply return a dendrogram. This may be useful if graphical parameters need to be calculated but rendering is not desired. (see value)

`c1.fun` (optional) Function to convert the vector of numeric values represented by `stat_col` to a character vector corresponding to colors. For dual channel plots, these colors may be combined with a second array of colors using by the method specified using the `combine_method` parameter. If not specified, `c1.fun` calculated automatically as a linear function.

`c2.fun` (optional) Same as c1.fun but for `stat_col_2`.

`geometry` (optional) Specifies either "horizontal" or "circular" type dendrogram plots. (default: horizontal)

`.plt.plot` (optional) Specifies the plot region of the output using `figure` coordinates, and excluding the legends. This can provide a greater degree of control for plotting, but most users will not need to adjust this. See the `plt` argument of the par graphics function for more information.

`leaves_pch` (optional) Used to specify the pch symbol used to represent dendrogram leaves. (default: 22 (open square), for horizontal dendrograms and dendextend version >= '1.16.0'; 15 (solid square) for horizontal dendrograms with dendextend version < '1.16.0', and for circular dendrograms, 16 (solid circle))

`leaf_char_shift` (optional) A parameter telling the function by how many character widths to shift the leaf labels. (default: 1)

`na.color` (optional) The color used for NA values. (default: "#CCCCCC")

`leaf_cex` (optional) The cex size of the leaf symbols. This is used when `n_col` is not specified, i.e. there is no leaf size scaling. (default: 1.5 * lab.cex)

`leaf_cex_range` (optional) The range of leaf sizes used in plots, from low to high. This is used when `n_col` is specified and leaf sizes are to be scaled. This may need to be reduced if leaves overlap or are clipped on one size. (default: c(0.5, 2.1))

`lab.cex` (optional) The cex size of dendrogram leaf labels (default: 0.9 * cex).

`tree_x_size.in` (optional) For horizontal dendrograms, this is the width of the dendrogram in inches, not including leaf labels, cluster brackets, or legends. (default: 2)

`legend_x_size.in` (optional) The width of legends in inches. (default: 2)

`left_margin.in` (optional) The width of the left margin in inches. Ignored if .plt.plot or .mai.plot is specifed. (default: 0)

`right_margin.in` (optional) The width of the right margin of the dendrogram in inches. Ignored if .plt.plot or .mai.plot is specifed. If unspecified, this is calculated automatically as `width - tree_x_size.in`.

`top_margin.in` (optional) The width of the top margin of the dendrogram in inches. Ignored if .plt.plot or .mai.plot is specifed. (default: if no `main` argument is specified, 0. If a `main` argument is specified, then it is calcualted as `cex.main * par('cin')[2] * mar.main`)

`bottom_margin.in` (optional) (optional) The width of the bottom margin in inches. Ignored if .plt.plot or .mai.plot is specifed. (default: 0)

`legend.downshift.in` (optional) Argument shifting the legend downward, in inches. This is useful for adjusting the alignment of the legend(s) with the top of the plot. (default: for horizontal dendrograms, 0; for circular dendrograms, 0.42)

`bkt_lmargin_chars` (optional) Width in character widths of the space between the leaf labels and the brackets indicating cluster/subnet groups. If the leaf labels need more space, this can be increased. (default: 4)

`legend_spacing.x.in` (optional) Space between plot and legend in inches. With some plot configurations, it may be useful to use negative values to bring the legends closer to the plot region. (default: 2 character widths)

`legend_spacing.y.in`
: (optional) Space between legends in inches. (default: 1 character height)

`legend.lab.cex`
: (optional) Legend x and y label size in cex. If unspecified, the function tries to pick a reasonable value based on available space.

`legend.axis.cex`
: (optional) Legend axis label size in cex. If unspecified, the function tries to pick a reasonable value based on available space.

`legend.free.cex.bool`
: (optional) Logical allowing independent optimized sizing of legend label font sizes if TRUE. (default: FALSE)

`main`
: (optional) Legend main title. (default: NULL)

`cex.main`
: (optional) Font size in cex units for the main title. (default: `1.35 * cex`)

`mar.main`
: (optional) Tells the function to reserve this many line heights for the main title. (default: 3.2)

`lines.main`
: (optional) Tells the function to place the main title this many lines away from the plot edge. (default: 1.5)

`colors.n`
: (optional) The number of colors per dimension of the color scale. For single channel plots, this will be equal to the number of colors in the color scale. For 2 channel plots, the number of colors is the square of this number. (default 100).

`legend.bg`
: (option) The color of the legend background. (default: `par('bg')`)

`legend.fg`
: (option) The color of the legend foreground. (default: `par('fg')`)

`draw.legend.box.bool`
: (option) Logical indicating whether bounding boxes should be drawn for the legends.

`DO_BROWSER`
: (option) Logical indicating whether browser() should be run for this function. (For debugging purposes, will probably remove.)

Details

Outputs of type pdf, png, and svg are supported for file outputs. File type is automatically detected from the file suffix, but can be overridden using the `out_format` argument.

Open symbols (with border and a fill color, `pch ( 21, 22, 23, 24, 25 )`) are used by default on dendextend versions < '1.16.0' for horizontal dendrograms. For earlier versions, and with circular dendrograms, open symbols are currently unsupported.

Value

An object of type 'dendrogram', with the attribute "GSNA_plot_params" containing a list of plot parameters. This list is useful for retrieving plot parameters set by the function, so that they might be optimized. Likewise, the dendrogram object itself can be replotted or analyzed by other means.

See Also

`gsnPareNetGenericHierarchic` `gsnPlotNetwork`

Examples

```
## Not run:
  gsnHierarchicalDendrogram( object  = analysis.GSN, pathways_title_col = NA )

## End(Not run)
```

gsnHierarchicalDendrogram.old
gsnHierarchicalDendrogram.old

Description

Generate a dendrogram plot of a hierarchical clustered set of GSNA distances.

Usage

```
gsnHierarchicalDendrogram.old(
  object,
  distance = NULL,
  subnet_colors = NULL,
  file = NULL,
  width = 7,
  height = NULL,
  mai = c(0, 0, 0, 5.6),
  cex = 0.7,
  subnetColorsFunction = gsnDendroSubnetColors_dark,
  id_col = NULL,
  id_nchar = NULL,
  pathways_title_col = c("Title", "Name", "NAME", "STANDARD_NAME"),
  substitute_id_col = NULL,
  font_face = "Courier",
  modules = NULL
)
```

Arguments

`object`	An object of the class `GSNData`
`distance`	(optional) A character vector of length one to indicate the desired distance metric to be used for generating a hierarchical dendrogram, e.g. 'lf', 'jaccard', 'stlf', etc. Defaults to the value of objects `default_distance`.
`subnet_colors`	(optional) A character vector of color codes matching the desired colors for subnets. If null then the colors are set automatically.
`file`	(optional) A file for outputting an SVG format file.
`width`	(optional) Number expressing output image width in inches, defaults to 7.
`height`	(optional) Number expressing output image height in inches, defaults to 0.16 times the number of gene sets.
`mai`	(optional) Margin size of the dendrogram in inches. It's set to allow plenty of space for the labels on the right side. (default `c(0,0,0,5.6)`)
`cex`	(optional) Font magnification parameter, passed to `stats:::plot.dendrogram()`
`subnetColorsFunction`	Function for generating subnet colors. This defaults to `gsnDendroSubnetColors_dark()`.
`id_col`	(optional) Character vector of length 1 indicating the name of the column to be used as an ID key in the pathways dataframe (or modules data if that is used, see below). This column should contain the same values as the names of the gene sets. This defaults to the value of the pathways id_col field.

id_nchar (optional) Integer indicating the number of characters to reserve in the dendrogram plot for the ID. If unspecified, it is equal to the maximal nchar of the specified ID (id_col or substitute_id_col).

pathways_title_col (optional) Character vector of length 1 indicating the name of the column in the pathways or modules data.frame to be used as a Title or descriptor in the plot. If not set the function looks for the following names: "Title", "Name", "NAME", "STANDARD_NAME", and takes the first that it finds. If set to NA, the title part of the label is suppressed.

substitute_id_col (optional) Character vector of length 1 indicating a column used to substitute an alternative ID for the labeling gene sets in data set. If set to NA, the ID in the plot is disabled.

font_face (optional) The font used for leaf labels, which should be a monospaced font like monospace or Courier for best results. (Default is Courier.)

modules (optional) Either a class tmod object containing MODULES annotation, or a data.frame also containing such data. This is to be used when a pathways data set is not available or insufficient for including the proper labels in plot.

Value

Invisibly returns a dendro object.

See Also

gsnPareNetGenericHierarchic gsnPlotNetwork

Examples

```
## Not run:
gsnHierarchicalDendrogram.old( object  = analysis.GSN, pathways_title_col = NA )

## End(Not run)
```

---

gsnImportCERNO *gsnImportCERNO*

Description

Add a CERNO^1 analysis pathways result set to a GSNData object. The data set can be either in the form of a data.frame or specified as import from a delimited text file.

Usage

```
gsnImportCERNO(
  object,
  pathways_data = NULL,
  filename = NULL,
  id_col = NULL,
  stat_col = NULL,
```

```
  sig_order = NULL,
  n_col = NULL,
  sep = "\t"
)
```

Arguments

object	A GSNData object.
pathways_data	An (optional) data.frame containing the results of CERNO analysis. (Either this or the filename argument must be set.
filename	An (optional) filename for data sets read from a text file containing CERNO results. This is ignored if the pathways_data argument is set.
id_col	(optional) A character vector of length 1 indicating the name of the column used as a key for gene sets or modules. This is normally the ID field of CERNO data, which must be the same as the names of gene sets specified in the tmod object or in the list of gene set vectors specified with the geneSetCollection argument used when building the gene set network. By default this value is 'ID', however if the user has added additional IDs to a CERNO results set, such as GO_ACCESSION, that can be specified here. The IDs must correspond to the names of the gene sets provided, or an error will be thrown.
stat_col	(optional) A character vector of length 1 indicating the name of the column used as a statistic to evaluate the quality of pathways results. By default, this is 'adj.P.val' for CERNO.
sig_order	(optional) Either 'loToHi' (default) or 'hiToLo' depending on the statistic used to evaluate pathways results.
n_col	(optional) Specifies the column containing the number of genes in the gene set. Generally, this is the number of genes in the gene set that are attested in an expression data set (Defaults to 'N1').
sep	A separator for text file import, defaults to "\t". Ignored if the filename argument is not specified.

Details

This method imports a CERNO^1 data set created by the tmod^2 package into a GSNData object.

Note: An error is thrown if all gene set IDs in the genePresenceAbsense are not present in the CERNO ID column. On the other hand, if there are gene set IDs present in the pathways data that are absent from the genePresenceAbsence matrix, then thes methods emit a warning. It also checks for the standard CERNO data set column names, and if some are missing, it will throw an error. They can still be imported via gsnImportGenericPathways.

Value

This returns a GSNData object containing imported pathways data.

References

1. Zyla J, Marczyk M, Domaszewska T, Kaufmann SHE, Polanska J, Weiner J. Gene set enrichment for reproducible science: comparison of CERNO and eight other algorithms. *Bioinformatics*. 2019;**35**: 5146–5154. doi:10.1093/bioinformatics/btz447
2. Weiner 3rd J, Domaszewska T. tmod: an R package for general and multivariate enrichment analysis. *PeerJ Preprints*; 2016 Sep. doi:10.7287/peerj.preprints.2420v1

See Also

gsnAddPathwaysData gsnImportGSEA gsnImportGSNORA gsnImportGenericPathways

Examples

```
## Not run:
gsn_object <- gsnImportCERNO( object = gsn_object, pathways_data = dat.cerno )

## End(Not run)
```

---

gsnImportDAVID *gsnImportDAVID*

Description

Add DAVID search data to a GSNData object, as generated by the the DAVID web application (https://david.ncifcrf.gov/https://david.ncifcrf.gov/) output using either the "Functional Annotation Chart" or "Functional Annotation Cluster" results output options. The data set can be either in the form of a data.frame or specified as import from an output text file. (See Details below)

Usage

```
gsnImportDAVID(
  object,
  pathways_data = NULL,
  filename = NULL,
  id_col = NULL,
  stat_col = NULL,
  sig_order = NULL,
  n_col = NULL,
  sep = "\t"
)
```

Arguments

object	A GSNData object.
pathways_data	An (optional) data.frame containing the results of DAVID analysis. (Either this or the filename argument must be set. Such a data.frame can be obtained by using the read_david_data_file() function to parse a DAVID "Functional Annotation Chart" or "Functional Annotation Cluster" results text file with the default options (output = "flat", redundant = FALSE, sep = "\t").
filename	An (optional) filename for data sets read from a text file containing DAVID results. This is ignored if the pathways_data argument is set.
id_col	(optional) A character vector of length 1 indicating the name of the column used as a key for gene sets or modules. This is normally the Term field of DAVID data, which must be the same as the names of gene sets in the gene set collection specified with the geneSetCollection argument used when building the gene set network. By default this value is 'Term'. The IDs must correspond to the names of the gene sets provided, or an error will be thrown.

stat_col	(optional) A character vector of length 1 indicating the name of the column used as a statistic to evaluate the quality of pathways results. The function scans through possible stat_col values ("FDR", "Bonferroni", "Benjamini", "PValue"), and uses the first one it finds.
sig_order	(optional) Either 'loToHi' (default) or 'hiToLo' depending on the statistic used to evaluate pathways results.
n_col	(optional) Specifies the column containing the number of genes in the gene set. Generally, this is the number of genes in the gene set that are attested in an expression data set (Defaults to 'Count', if that is present, otherwise
sep	A separator for text file import, defaults to "\t". Ignored if filename is not specified.

Details

Note: An error is thrown if all gene set IDs in the genePresenceAbsense are not present in the GSEA NAME column. However, if there are gene set IDs present in the pathways data that are absent from the $genePresenceAbsence matrix, then this method emits a warning. It also checks for the standard GSEA data set column names, and if some are missing, it will emit a warning.

Value

This returns a GSNData object containing imported pathways data.

See Also

gsnAddPathwaysData gsnImportCERNO gsnImportGSNORA gsnImportGenericPathways

Examples

```
## Not run:
gsn_object <- gsnImportDAVID( object = gsn_object, pathways_data = dat.david )

## End(Not run)
```

---

gsnImportGenericPathways

gsnImportGenericPathways

Description

Import a data.frame or text file containing a pathways result set to a GSNData object. The id_col and stat_col should be specified, but if they are not, the function attempts to guess.

Usage

```
gsnImportGenericPathways(
  object,
  pathways_data = NULL,
  filename = NULL,
  type = "generic",
```

```
  id_col = NULL,
  stat_col = NULL,
  stat_col_2 = NULL,
  sig_order = NULL,
  sig_order_2 = NULL,
  n_col = NULL,
  sep = "\t"
)
```

Arguments

object	A GSNData object.
pathways_data	An (optional) data.frame containing the pathways analysis. (Either this or the filename argument must be set.
filename	An (optional) filename for data sets read from a text file containing pathways results. This is ignored if the pathways_data argument is set.
type	A cbaracter vector of length 1 indicating the type of result set. This defaults to 'generic'.
id_col	(optional) A character vector of length 1 indicating the name of the column used as a key for gene sets or modules. This should be the same as the set of names of gene sets in the gene set collection specified by the geneSetCollection argument used in building gene set networks. If not specified, the function will search for "ID", "id", "NAME" & "Term" in the data set's column names, in that order, taking the first one it finds. The values in the column must correspond to the names of the gene sets provided, or an error will be thrown.
stat_col	(optional) A character vector of length 1 indicating the name of the column used as a statistic to evaluate the quality of pathways results. If unspecified, the function uses regular expressions to search for a column that is labeled as a p-value or p-adj.
stat_col_2	(optional) A character vector of length 1 indicating the name of the column used as an optional second statistic to evaluate the quality of pathways results. If unspecified, the value is NULL.
sig_order	(optional) Either 'loToHi' (default) or 'hiToLo' depending on the statistic used to evaluate pathways results.
sig_order_2	(optional) Either 'loToHi' (default) or 'hiToLo' depending on the stat_col_2 statistic used to evaluate pathways results.
n_col	(optional) The name of a pathways data column that contains gene set size information. If unset, the function will scan for the strings 'N1', 'N', 'SIZE', and 'Count', taking the fist one it finds.
sep	A separator for text file import, defaults to "\t". Ignored if the filename argument is not specified.

Value

This returns a GSNData object containing imported pathways data.

Note: An error is thrown if all gene set IDs in the $genePresenceAbsence field are not present in the GSNORA ID column. On the other hand, if there are gene set IDs present in the pathways data that are absent from the genePresenceAbsence matrix, then thes methods emit a warning. It also checks for the standard GSNORA data set column names, and if some are missing, it will throw an error.

See Also

gsnAddPathwaysData gsnImportCERNO gsnImportGSEA gsnImportGenericPathways

Examples

```
## Not run:
gsn_object <- gsnImportGenericPathways( object = gsn_object,
                                        pathways_data = dat.cerno,
                                        n_col = "N1.Condition1",
                                        stat_col = "adj.P.Val.Treat1".
                                        sig_order = "loToHi",
                                        stat_col_2 = "adj.P.Val.Treat2".
                                        sig_order_2 = "loToHi"
                                         )

## End(Not run)
```

---

gsnImportGSEA *gsnImportGSEA*

Description

Add GSEA search data to a GSNData object, as generated by the the GSEA package. The data set can be either in the form of a data.frame or specified as import from a delimited text file. (See Details below)

Usage

```
gsnImportGSEA(
  object,
  pathways_data = NULL,
  filename = NULL,
  id_col = NULL,
  stat_col = NULL,
  sig_order = NULL,
  n_col = NULL,
  sep = "\t"
)
```

Arguments

object	A GSNData object.
pathways_data	An (optional) data.frame containing the results of GSEA analysis. (Either this or the filename argument must be set.
filename	An (optional) filename for data sets read from a text file containing GSEA results. This is ignored if the pathways_data argument is set.
id_col	(optional) A character vector of length 1 indicating the name of the column used as a key for gene sets or modules. This is normally the NAME field of GSEA data, which must be the same as the names of gene sets specified in the tmod object or in the list of gene set vectors specified with the geneSetCollection argument

used when building the gene set network. By default this value is 'NAME'. The IDs must correspond to the names of the gene sets provided, or an error will be thrown. **NOTE:** In the tmod::tmodImportMSigDB function provided by the tmod package, the default ID is an MSigDB accession, but GSEA data sets do not use this accession. The NAME column used in GSEA results set corresponds instead to the STANDARD_NAME field in the MSigDB XML database file. This STANDARD_NAME field is not preserved by the standard tmod::tmodImportMSigDB utility function, but instead reformatted converting underscores to spaces and non-initial letters to lower case. Therefore, when using GSEA data sets with an MSigDB gene set collection imported using tmod::tmodImportMSigDB the NAME fields need to be mapped to the ID or vice versa.

stat_col (optional) A character vector of length 1 indicating the name of the column used as a statistic to evaluate the quality of pathways results. The function scans through possible stat_col values ("FDR q-val", "FDR.q.val", "FWER p-val", "FWER.p.val", "NOM p-val", "NOM.p.val"), and uses the first one it finds. (The presence of spaces and hypens in the column names necessitates flexibility here. Depending on how GSEA results sets are read in, spaces and hyphens may be substituted with periods.)

sig_order (optional) Either 'loToHi' (default) or 'hiToLo' depending on the statistic used to evaluate pathways results.

n_col (optional) Specifies the column containing the number of genes in the gene set. Generally, this is the number of genes in the gene set that are attested in an expression data set (Defaults to 'SIZE').

sep A separator for text file import, defaults to "\t". Ignored if filename is not specified.

Details

GSEA results directories generally contain files with names beginning with gsea_report_for_ and with the .xls suffix. This method is designed to handle such data sets.

Note: An error is thrown if all gene set IDs in the genePresenceAbsense are not present in the GSEA NAME column. However, if there are gene set IDs present in the pathways data that are absent from the $genePresenceAbsence matrix, then this method emits a warning. It also checks for the standard GSEA data set column names, and if some are missing, it will emit a warning.

Value

This returns a GSNData object containing imported pathways data.

See Also

gsnAddPathwaysData gsnImportCERNO gsnImportGSNORA gsnImportGenericPathways

Examples

```
## Not run:
gsn_object <- gsnImportGSEA( object = gsn_object, pathways_data = dat.cerno )

## End(Not run)
```

gsnImportGSNORA *gsnImportGSNORA*

Description

Add GSNORA search data to a GSNData object, as generated by the gsnORAtest function in this package. The data set can be either in the form of a data.frame or specified as import from a delimited text file.

Usage

```
gsnImportGSNORA(
  object,
  pathways_data = NULL,
  filename = NULL,
  id_col = NULL,
  stat_col = NULL,
  sig_order = NULL,
  n_col = NULL,
  sep = "\t"
)
```

Arguments

object	A GSNData object.
pathways_data	An (optional) data.frame containing the results of GSNORA analysis. (Either this or the filename argument must be set.
filename	An (optional) filename for data sets read from a text file containing GSNORA results. This is ignored if the pathways_data argument is set.
id_col	(optional) A character vector of length 1 indicating the name of the column used as a key for gene sets or modules. This is normally the ID field of GSNORA data, which must be the same as the names of gene sets specified in the tmod object or in the list of gene set vectors specified with the geneSetCollection argument used when building the gene set network. By default this value is 'ID', however if the user has added additional IDs to a CERNO results set, such as GO_ACCESSION, that can be specified here. The IDs must correspond to the names of the gene sets provided, or an error will be thrown.
stat_col	(optional) A character vector of length 1 indicating the name of the column used as a statistic to evaluate the quality of pathways results. By default, this is 'adj.P.1S' for GSNORA.
sig_order	(optional) Either 'loToHi' (default) or 'hiToLo' depending on the statistic used to evaluate pathways results.
n_col	(optional) Specifies the column containing the number of genes in the gene set. Generally, this is the number of genes in the gene set that are attested in an expression data set (Defaults to 'N').
sep	A separator for text file import, defaults to "\t". Ignored if the filename argument is not specified.

Value

This returns a GSNData object containing imported pathways data.

Note: An error is thrown if all gene set IDs in the genePresenceAbsense are not present in the GSNORA ID column. On the other hand, if there are gene set IDs present in the pathways data that are absent from the genePresenceAbsence matrix, then thes methods emit a warning. It also checks for the standard GSNORA data set column names, and if some are missing, it will throw an error.

See Also

gsnAddPathwaysData gsnImportCERNO gsnImportGSEA gsnImportGenericPathways

Examples

```
## Not run:
gsn_object <- gsnImportGSNORA( object = gsn_object, pathways_data = dat.cerno )

## End(Not run)
```

---

gsnMergePathways *gsnMergePathways*

Description

Merge pathways data and subnets into a data.frame that includes subnet assignment and intra-subnet rank.

Usage

```
gsnMergePathways(
  object,
  pathways.data = NULL,
  distance = NULL,
  id_col = NULL,
  stat_col = NULL,
  sig_order = NULL
)
```

Arguments

object	A GSNData object upon which gsnAssignSubnets() has been called.
pathways.data	(optiional) data.frame containing a pathways results. Not necessary if pathways data have already been imported.
distance	(optional) character vector of length 1 indicating which set of subnets to be used if the GSNData object contains subnets derived from more than one distance matrix.
id_col	(optional) ID column to be used for merging subnets. Defaults to the value of id_col already set during import of pathways data, if that has already been done.

`stat_col` (optional) The name of the column column containing the statistic to be used for ordering subnets and performing intra-subnet ranking. Defaults to the value of `stat_col` already set during import of pathways data, if that has already been done.

`sig_order` (optional) Character vector of length 1 indicating the whether low values of the statistic are most significant ("loToHi", the default) or high values ("hiToLo") for ordering subnets and performing. Defaults to the value of `sig_order` already set during import of pathways data, if that has already been done.

Details

In the standard workflow, just the object parameter is generally necessary. If subnets have been calculated for multiple distance matrices and the subnets desired are not associated with the current default distance, then the `distance` parameter can be specified.

Value

A data.frame containing pathways data with merged subnet assignments and subnetRank values.

See Also

`gsnAddPathwaysData()` `gsnImportCERNO()` `gsnImportGSNORA()`, `gsnImportGSEA()` `gsnImportGenericPathways()`

Examples

```
## Not run:
analysis.mergePathways <- gsnMergePathways( object = analysis.GSN )

## End(Not run)
```

`gsnORAtest` *gsnORAtest*

Description

Perform an ORA test using an experimentally-derived gene set to query a gene set collection.

Usage

```
gsnORAtest(l, bg, geneSetCollection, Alpha = 0.05, full = FALSE)
```

Arguments

`l` A vector containing an experimentally-derived set of genes. These may be significantly differentially expressed genes, genes with differential chromatin accessability or positives from a screen.

`bg` A vector containing a background of observable genes.

`geneSetCollection` A gene set collection to query, either a tmod object or a list of character vectors containing gene sets for which the list element names are the gene set IDs.

`Alpha`	The alpha value setting the significance cutoff adjusted p-value.
`full`	This gives additional data in the results set, specifically the contingency table values.

Details

This function is provided to allow rapid and easy overrepresentation analysis using an unordered experimental gene set to query a gene set collection that may be either an arbitrary list of gene-sets, or an `tmod` class gene set collection. The statistical tests provided include both the standard two-sided Fisher and a 1-sided Fisher test, similar to what is provided by the DAVID pathways analysis web application^2.

If a list of gene sets is provided as the `geneSetCollection` argument, it must be structured as a list of character vectors containing gene symbols (or whatever identifiers are used for the supplied experimental gene set),

Value

Returns a data.frame with an ORA (overrepresentation analysis) results set containing the following columns:

- *ID*: the gene set identifiers.
- *Title*: The "Title" field from `tmod` class gene set collection objects, corresponding to the reformatted `STANDARD_NAME` field in an MSigDB xml file, with spaces substituted for underscores and initial only uppercase. **NOTE:** If the search is done using a list of gene sets rather than a `tmod` object, this column will contain NA.
- *a*: the number of genes observed in the background but not in *l* or the queried gene set. (present only if `full == TRUE`)
- *b*: the number of observed genes in *l* but not the queried gene set. (present only if `full == TRUE`)
- *c*: the number of observed genes in the queried gene set but not *l*. (present only if `full == TRUE`)
- *d*: the number of observed genes in both *l* and the queried gene set, i.e. the overlap. (present only if `full == TRUE`)
- *N*: the number of observed genes the queried gene set.
- *Enrichment*: The fold overrepresentation of genes in the overlap set *d* calculated as: `E = (d / (c+d)) / ((b+d)/(a+b+c+d))`
- *P_2S*: 2-sided Fisher *p*-value. (*NOT* log-transformed, present only if `full == TRUE`)
- *adj.P.2S*: 2-sided Fisher *p*-value corrected using the method of Benjamini & Hochberg^1 and implemented in the `stats` package. (present only if `full == TRUE`)
- *P_1S*: 1-sided Fisher *p*-value. (*NOT* log-transformed.)
- *adj.P.1S*: 1-sided Fisher *p*-value corrected using the method of Benjamini & Hochberg^1 and implemented in the `stats` package. (present only if `full == TRUE`)

References

1. Benjamini, Y., and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series B*, **57**, 289–300. http://www.jstor.org/stable/2346101.
2. Dennis G Jr, Sherman BT, Hosack DA, Yang J, Gao W, Lane HC, Lempicki RA. (2003). DAVID: Database for Annotation, Visualization, and Integrated Discovery. *Genome Biol.*, **4**(5):P3. Epub 2003 Apr 3.

See Also

gsnORAtest_cpp p.adjust

Examples

```
## Not run:
results.ORA <- gsnORAtest(l = DE_GENES_UP,
                          bg = ALL_GENES_IN_DATASET,
                          geneSetCollection = msig.tmod )

## End(Not run)
```

---

gsnORAtest_cpp *gsnORAtest_cpp*

Description

This function performs ORA analysis and returns a data.frame containing various statistics including fold enrichment, and 1 and 2-tailed p-values. (see details)

Usage

```
gsnORAtest_cpp(l, bg, geneSetCollection)
```

Arguments

l	(required) A character vector containing a list of gene identifiers. These are generally differentially expressed genes either genes significantly up or significantly down, but they can also be a list of genes that camme out of a genetic screen, gene loci with differential chromatin accessibility generated by ATACSeq data, lists of genes from GWAS, etc. The order of the genes is unimportant.
bg	(required) A character vector containing a list of gene identifiers corresponding to the total background of observable genes.
geneSetCollection	(required) A list of gene sets, in which the gene sets are character vectors containing gene symbols, and the list names are the corresponding gene set identifiers. NOTE: This must be a list, not a tmod object. It is trivial to extract such a list from a tmod object, however. The $MODULES2GENES field of the tmod object contains a suitable list.

Details

This is the main workhorse function for the ORA test in the GSNA package, however, it performs no filtering of the output data set, nor *p*-value adjustment, and most users of the package will want to use gsnORAtest() function instead, which calculates adjusted *p*-values, filters the output data for significance, and can include a Title field in the output data.frame.

Value

A data frame containing the results of overrepresentation analysis.

- *ID*: the gene set identifiers.
- *a*: the number of genes observed in the background but not in *l* or the queried gene set.
- *b*: the number of observed genes in *l* but not the queried gene set.
- *c*: the number of observed genes in the queried gene set but not *l* and
- *d*: the number of observed genes in both *l* and the queried gene set, i.e. the overlap.
- *N*: the number of observed genes the queried gene set.
- *Enrichment*: The fold overrepresentation of genes in the overlap set *d* calculated as: `E = (d / (c+d)) / ((b+d)/(a+b+c+d))`
- *P_2S*: 2-sided Fisher *p*-value. (*NOT* log-transformed.)
- *P_1S*: 1-sided Fisher *p*-value. (*NOT* log-transformed.)

See Also

`gsnORAtest`

`gsnParedVsRawDistancePlot`

gsnParedVsRawDistancePlot

Description

A method for generating a bivariate plot of pared/scaled distances vs. raw distances.

Usage

```
gsnParedVsRawDistancePlot(object, distance = NULL, ...)
```

Arguments

`object`	An object of type `GSNData` containing a distance matrix.
`distance`	(optional) character vector of length 1 indicating which pared distance matrix is to be used for assigning subnets. This defaults to the 'default_distance'.
`...`	Additional graphical parameters to be passed to `plot.default()`.

See Also

`gsnPareNetGenericHierarchic`

Examples

```
## Not run:
 gsnParedVsRawDistancePlot( object = analysis.GSN, col = "blue" )

## End(Not run)
```

gsnPareNetGenericHierarchic *gsnPareNetGenericHierarchic*

Description

Method to perform hierarchical clustering and paring of gene set networks.

Usage

```
gsnPareNetGenericHierarchic(
  object,
  distance = NULL,
  extreme = NULL,
  cutoff = NULL,
  keepOrphans = TRUE,
  matrix_scaling_fun = NULL,
  lower_is_closer = NULL,
  k = NULL,
  h = NULL,
  method = "average"
)
```

Arguments

object	An object of type GSNData containing a distance matrix.
distance	(optional) character vector of length 1 indicating which pared distance matrix is to be used for assigning subnets. This defaults to the 'default_distance'.
extreme	(optional) Either min or max indicating whether low or high values are most significant, i.e. to be interpreted as the shortest distance for nearast neighbor paring. This defaults to the value set for the optimal_extreme field of the specified distance matrix.
cutoff	(optional) A cutoff specifying a maximal of minimal value that will be retained, dependent on the distance metric being used. This is not usually necessary to specify for hierachical clustering. (see details)
keepOrphans	A boolean indicating whether 'orphan' gene sets that have no nearest neighbors should be retained in the final network. (default TRUE)
matrix_scaling_fun	A function to perform transformation and scaling of the distance matrix. The default, distMat2UnitNormRank converts the distance matrix to ranks and scales the resulting numbers to a range between 0 and 1. If set to NULL, the distances are not scaled or transformed. (see details)
lower_is_closer	Boolean indicating that lower values should be treated as closer for the sake of hierarchical clustering.
k	(optional) Parameter passed to cutree to determine the number of desired clusters. If both k and h are NULL, a value for k will be chosen. (see details)
h	(optional) Parameter passed to cutree to determine the cutting height for breaking the clusters into groups. (see details)

`method` (optional) Parameter passed to `hclust()` to specify the hierarchical clustering method used. (default "average")

Details

This method performs hierarchical clustering, then joins the members of each cluster. This joining occurs as follows:

1. First, only the edges between gene sets belonging to the same hierarchical cluster are considered, and the edges within each cluster are ordered by distance.
2. The first edge is the edge defined by the shortest distance.
3. Subsequent edges are added to the subnet by selecting the shortest from the edges shared by one joined and one unjoined gene set.
4. This process is repeated until all gene sets in a cluster are joined as a subnet.

This joining method differs from nearest neighbor joining in that unjoined nodes are initially joined, not to their nearest neighbor necessarily, but to their nearest neighbor from among the nodes already joined together in a subnet. This method avoids bifurcation of subnets that could occur by regular nearest neighbor joining.

NOTE: The `matrix_scaling_fun` argument is a function that takes the distance matrix and transforms it into scaled data appropriate for hierarchical clustering. (As such, it should return data with low values indicating closer gene sets, as opposed to a Jaccard index where high values are closest.) Because this function may transform the data from a scale where high values are close to one where low values are close, such functions should return a matrix with a `lower_is_closer` attribute set as `TRUE` to indicate that. If the `lower_is_closer` attribute is not set by `matrix_scaling_fun`, then it will be assumed to be the same as the raw distance matrix, which may generate an error if the `optimal_extreme` of the distance matrix is not `'min'`. This value will be used to set the corresponding `$distances[[distance]]$pared_optimal_extreme` field in the GSNData object. In general, a scaling transformation is necessary because some potential distance metrics are in log-space and have skewed distributions and negative values (like log Fisher) or are actually similarity metrics, with higher values being closer. In this way they differ from standard distances, and require transformation to be suitable for hierarchical clustering. The default, `matrix_scaling_fun` argument, `distMat2UnitNormRank()` scales the data to a range between 0 and 1, and converts it to a uniform distribution. This may be a bit extreme for some purposes, but it allows the hierarchical clustering method to work simply with default values for most users obviating the need to transform the data or adjust default parameters in many cases. For a plot of the relationship between the raw and transformed/scaled pared distances, see `gsnParedVsRawDistancePlot`.

Value

A `GSNData` copy of the original `object` argument containing a pared distance matrix for the specified distance metric.

See Also

`gsnPareNetGenericToNearestNNeighbors` `distMat2UnitNormRank` `gsnParedVsRawDistancePlot`

Examples

```
## Not run:
analysis.GSN <- gsnPareNetGenericHierarchic( object = analysis.GSN,
                                             distance = "lf",
                                             lower_is_closer = TRUE )
```

```
# To perform hierarchical clustering without scaling or transforming the distance matrix:
analysis_unscaled.GSN <- gsnPareNetGenericHierarchic( object = analysis.GSN,
                                                      distance = "custom_dist",
                                                      matrix_scaling_fun = NULL,
                                                      lower_is_closer = TRUE )

## End(Not run)
```

---

gsnPareNetGenericHierarchic.old
gsnPareNetGenericHierarchic.old

Description

Method to perform hierarchical clustering and paring of gene set networks.

Usage

```
gsnPareNetGenericHierarchic.old(
  object,
  distance = NULL,
  extreme = NULL,
  cutoff = NULL,
  keepOrphans = TRUE,
  matrix_scaling_fun = distMat2UnitNormRank,
  lower_is_closer = NULL,
  k = NULL,
  h = NULL,
  method = "average"
)
```

Arguments

object	An object of type GSNData containing a distance matrix.
distance	(optional) character vector of length 1 indicating which pared distance matrix is to be used for assigning subnets. This defaults to the 'default_distance'.
extreme	(optional) Either min or max indicating whether low or high values are most significant, i.e. to be interpreted as the shortest distance for nearast neighbor paring. This defaults to the value set for the optimal_extreme field of the specified distance matrix.
cutoff	(optional) A cutoff specifying a maximal of minimal value that will be retained, dependent on the distance metric being used. This is not usually necessary to specify for hierachical clustering. (see details)
keepOrphans	A boolean indicating whether 'orphan' gene sets that have no nearest neighbors should be retained in the final network. (default TRUE)
matrix_scaling_fun	A function to perform transformation and scaling of the distance matrix. The default, distMat2UnitNormRank converts the distance matrix to ranks and scales the resulting numbers to a range between 0 and 1. If set to NULL, the distances are not scaled or transformed. (see details)

lower_is_closer
: Boolean indicating that lower values should be treated as closer for the sake of hierarchical clustering.

k
: (optional) Parameter passed to cutree to determine the number of desired clusters. If both k and h are NULL, a value for k will be chosen. (see details)

h
: (optional) Parameter passed to cutree to determine the cutting height for breaking the clusters into groups. (see details)

method
: (optional) Parameter passed to `hclust()` to specify the hierarchical clustering method used. (default "average")

Details

This method performs hierarchical clustering, then joins the members of each cluster. This joining occurs as follows:

1. First, only the edges between gene sets belonging to the same hierarchical cluster are considered, and the edges within each cluster are ordered by distance.
2. The first edge is the edge defined by the shortest distance.
3. Subsequent edges are added to the subnet by selecting the shortest from the edges shared by one joined and one unjoined gene set.
4. This process is repeated until all gene sets in a cluster are joined as a subnet.

This joining method differs from nearest neighbor joining in that unjoined nodes are initially joined, not to their nearest neighbor necessarily, but to their nearest neighbor from among the nodes already joined together in a subnet. This method avoids bifurcation of subnets that could occur by regular nearest neighbor joining.

NOTE: The `matrix_scaling_fun` argument is a function that takes the distance matrix and transforms it into scaled data appropriate for hierarchical clustering. (As such, it should return data with low values indicating closer gene sets, as opposed to a Jaccard index where high values are closest.) Because this function may transform the data from a scale where high values are close to one where low values are close, such functions should return a matrix with a `lower_is_closer` attribute set as `TRUE` to indicate that. If the `lower_is_closer` attribute is not set by `matrix_scaling_fun`, then it will be assumed to be the same as the raw distance matrix, which may generate an error if the `optimal_extreme` of the distance matrix is not `'min'`. This value will be used to set the corresponding `$distances[[distance]]$pared_optimal_extreme` field in the GSNData object. In general, a scaling transformation is necessary because some potential distance metrics are in log-space and have skewed distributions and negative values (like log Fisher) or are actually similarity metrics, with higher values being closer. In this way they differ from standard distances, and require transformation to be suitable for hierarchical clustering. The default, `matrix_scaling_fun` argument, `distMat2UnitNormRank()` scales the data to a range between 0 and 1, and converts it to a uniform distribution. This may be a bit extreme for some purposes, but it allows the hierarchical clustering method to work simply with default values for most users obviating the need to transform the data or adjust default parameters in many cases. For a plot of the relationship between the raw and transformed/scaled pared distances, see `gsnParedVsRawDistancePlot`.

Value

A `GSNData` copy of the original `object` argument containing a pared distance matrix for the specified distance metric.

See Also

`gsnPareNetGenericToNearestNNeighbors` `distMat2UnitNormRank` `gsnParedVsRawDistancePlot`

Examples

```
## Not run:
analysis.GSN <- gsnPareNetGenericHierarchic( object = analysis.GSN,
                                             distance = "lf",
                                             lower_is_closer = TRUE )

# To perform hierarchical clustering without scaling or transforming the distance matrix:
analysis_unscaled.GSN <- gsnPareNetGenericHierarchic( object = analysis.GSN,
                                                      distance = "custom_dist",
                                                      matrix_scaling_fun = NULL,
                                                      lower_is_closer = TRUE )

## End(Not run)
```

`gsnPareNetGenericToNearestNNeighbors`
gsnPareNetGenericToNearestNNeighbors

Description

General method to pare GSNData distance matrices to nearest neighbor subset, applying any low or high value cutoffs that may be required.

Usage

```
gsnPareNetGenericToNearestNNeighbors(
  object,
  distance = NULL,
  extreme = NULL,
  cutoff = 0,
  keepOrphans = TRUE,
  N = 1
)
```

Arguments

`object`	An object of type `GSNData` containing a distance matrix.
`distance`	(optional) character vector of length 1 indicating which pared distance matrix is to be used for assigning subnets. This defaults to the 'default_distance'.
`extreme`	(optional) Either `min` or `max` indicating whether low or high values are most significant, i.e. to be interpreted as the shortest distance for nearast neighbor paring. This defaults to the value set for the `optimal_extreme` field of the specified `distance` matrix.
`cutoff`	(optional) A cutoff specifying a maximal of minimal value that will be retained, dependent on the distance metric being used. The default value is 0, but this is likely incorrect for most purposes. For 'lf' and 'stlf' distances, we recommend a value of -90. For 'jaccard' distances, we recommend 0.3-0.4. (see details)
`keepOrphans`	A boolean indicating whether 'orphan' gene sets that have no nearest neighbors should be retained in the final network. (default `TRUE`)
`N`	Integer indicating the number of nearest neighbors to retain. (default 1)

Details

This method pares the GSN networks down to N nearest neighbors, with several tunable parameters. It is generally useful to include a cutoff for this method to remove weak associations between gene sets, but this is heavily dependent on the distance metric being used. A histogram or density plot showing the distribution of `raw` distances may be useful for determining a suitable value, since inflection points can guide selection of this cutoff. Such a plot may be generated using the `gsnDistanceHistogram()` method.

An alternative to this paring method is hierachical clustering implemented in the `gsnPareNetGenericHierarchic` method.

Value

A GSNData object containing a pared distance matrix for the specified `distance` metric.

See Also

`gsnPareNetGenericHierarchic` `gsnDistanceHistogram`

Examples

```
## Not run:
analysis.GSN <- gsnPareNetGenericToNearestNNeighbors( object = analysis.GSN,
                                                      distance = "lf",
                                                      cutoff = -90 )

## End(Not run)
```

`gsnPlotNetwork` *gsnPlotNetwork*

Description

Function for plotting the networks within GSNData objects.

Usage

```
gsnPlotNetwork(
  object,
  pathways_dat = NULL,
  distance = NULL,
  id_col = NULL,
  substitute_id_col = NULL,
  stat_col = NULL,
  stat_col_2 = NULL,
  sig_order = NULL,
  sig_order_2 = NULL,
  n_col = NULL,
  optimal_extreme = NULL,
  transform_function = nzLog10,
  pathways_title_col = c("Title", "Name", "NAME", "STANDARD_NAME"),
```

```
 edge_colors = c("black", "purple", "blue", "green", "yellow4", "orange", "red"),
  vertex_colors = c("white", "yellow", "red"),
  vertex_colors.1 = c("white", "red"),
  vertex_colors.2 = c("white", "blue"),
  combine_method = "scaled_geomean",
  na.color = "#CCCCCC",
  filename = NULL,
  out_format = NULL,
  width = NULL,
  height = NULL,
  vertex.shape = "circle",
  vertex.size = NULL,
  vertex.size.range = NULL,
  vertex.label.cex = NULL,
  vertex.label.col = NULL,
  vertex.frame.color = par("fg"),
  contrasting_color.fun = NULL,
  scale_labels_by_vertex = TRUE,
  max_edge_width = NULL,
  scale.edges.by.distance = FALSE,
  color.edges.by.distance = FALSE,
  edge_arrow_size = NULL,
  seed = 29189892,
  layout = function(x) {
     igraph::layout_with_fr(x, grid = "nogrid")
 },
  .plot = igraph::plot.igraph,
  show.legend = TRUE,
  legend.lab.cex = NULL,
  legend.axis.cex = NULL,
  legend.fg = par("fg"),
  legend.bg = "#DDDDDD",
  legend.vertex.fg = NULL,
  legend.vertex.bg = "#DDDDDD",
  font_face = par("family"),
  main = NULL,
  cex.main = NULL,
  mar.main = 3.2,
  lines.main = 0.9,
  .mar.plot = NULL,
  draw.legend.box.bool = FALSE,
  legend.free.cex.bool = FALSE,
  legend_x_size.in = NULL,
  colors.n = 100,
  new = FALSE,
  legend_spacing.x.in = 2 * par("cin")[1],
  legend_spacing.y.in = par("cin")[2],
  DO_BROWSER = FALSE
)
```

Arguments

object	A GSNData object containing a pared distance matrix with edges. NOTE: when calling as plot.GSNData, use the argument x instead.
pathways_dat	(optional) data.frame containing associated pathways data. This defaults to whatever pathways data has already been imported into this GSNData object in object$pathways$data.
distance	(optional) The name of a distance metric used, defaults to whatever default_distance is.
id_col	(optional) This is the name of the column in the pathways data.frame that corresponds to the names of gene sets. The default value is specified by object$pathways$id_col. (See details.)
substitute_id_col	(optional) This is the name of the column that is to be substituted for the id_col column when labeling network vertices. (See details.)
stat_col	(optional) This is the name of the column in the pathways data.frame that contains a significance value for coloring network vertices. The default value is specified by object$pathways$stat_col.
stat_col_2	(optional) This is the name of an optional second column in the pathways data.frame that contains a significance value for coloring network vertices in a 2-color network. The default value is specified by object$pathways$stat_col_2. When specified, a 2-color network is generated. To force a 2-color network to plot as a standard 1-color network using stat_col alone, use stat_col_2 = NA.
sig_order	(optional) This indicates the behavior of stat_col, whether low values ('loToHi') or high values ('hiToLo') are most significant. The default value is specified in object$pathways$sig_order.
sig_order_2	(optional) This indicates the behavior of stat_col_2, whether low values ('loToHi') or high values ('hiToLo') are most significant. The default value is specified in object$pathways$sig_order_2.
n_col	(optional) This is the name of the column in the pathways data.frame that contains a value for gene set size, or any other value intended to be the bases of leaf scaling. When specified, leaf sizes will be scaled by this value, either as a function argument, or in the object$pathways$n_col field. An NA value can be used to override the the value in object$pathways$n_col and suppress leaf scaling when n_col has been set in the object. (default is the value in object$pathways$n_col).
optimal_extreme	(optional) This indicates the behavior of the statistic used to generate the distance metric, specifically whether low values ('min') or high values 'max' are to be regarded as close. This is used for scaling the width and the color of the edges connecting vertices. See scale.edges.by.distance, below: (default: object$distances[distance]$pared_optimal_extreme or if that's NULL, object$distances[distance]$optimal_extreme)
transform_function	(optional) This is a function to transform the values in stat_col so that they are suitable for amenable to color-scaling. For *p*-values, a log transformation is often useful, but can produce negative infinities if the transformation is applied to zero. By default the function is the nzLog10 (non-zero log10) function, provided by this package, which adds a small pseudocount to p-values when log10 transforming values equal to zero. If values in stat_col are less than

zero, then log10 transformation is inappropriate and will introduce NAs, and therefore some other method should be used. (default: nzLog10)

pathways_title_col
: (optional) Indicates a column to be used as the 'Title' column for network vertices. If unset, the function attempts to search for a title column from the following values: c("Title", "Name", "NAME", "STANDARD_NAME") (See details.)

edge_colors
: (optional) A vector of colors included to generate a scale represent the numerical value of the edge distances. By default, the colors are arranged as a rainbow with black and purple representing the greatest distannces, and orange and red the nearest distances. This feature (and argument) will likely be deprecated in future versions. (default: edge_colors = c("black", "purple", "blue", "green","yellow4", "orange","red"))

vertex_colors
: (optional) This is the standard set of colors used for a standard single color network. By default, c("white","yellow","red") is used, coloring low values white, high values red, and intermediate values yellow if sig_order is "loToHi" and vice versa if sig_order is "hiToLo".

vertex_colors.1
: (optional) This is the range of colors used for a 2-color network corresponding to values of stat_col. Up to 2 colors can be used, and should correspond to a color contrasting with vertex_colors.2. The default is c("white","red"), coloring high values red and low values white if sig_order is "loToHi" and vice versa if sig_order is "hiToLo".

vertex_colors.2
: (optional) This is the range of colors used for a 2-color network corresponding to values of stat_col_2. Up to 2 colors can be used, and should correspond to a color contrasting with vertex_colors.2. The default is c("white","blue"), coloring high values blue and low values white if sig_order_2 is "loToHi" and vice versa if sig_order is "hiToLo".

combine_method
: (optional) For dual channel plots this is a string used to indicate how colors are combined to generate a two dimensional color scale. Options are "scaled_geomean" (same as "default"), "standard" (same as "euclidean"), "negative_euclidean", "mean", and "additive". See details.

na.color
: (optional) This color is assigned to vertices for which there is an NA value. (default: "#CCCCCC")

filename
: (optional) An output file name for the plot. If 'out_format' is not set (see below), the output file type will be determined by the file suffix, which can be '.svg', '.pdf', or '.png'. If the out_format cannot be determined from the file name, than it may be manually set with out_format. If the output file type cannot be determined from the filename or out_format arguments, an error will be thrown.

out_format
: (optional) Output filetype when filename is specified, either 'svg', 'png', 'pdf', or 'plot' (default if filename is not specified). For more information, see Details.

width
: (optional) Sets the width of the output canvas in inches. Defaults to the width of the present graphical device.

height
: (optiona) Sets the height of the output canvas in inches. Defailts to the height of the present graphical device.

vertex.shape
: (optional) Shape of the vertex, passed to igraph::plot.igraph. By default, the value is 'circle'.

vertex.size (optional) Size of vertices, passed to igraph::plot.igraph. By default, the value is NULL, and the function attempts to pick a reasonable value based on the canvas size and the number of gene sets.

vertex.size.range
(optional) The range of vertex sizes used in plots, from low to high. This is used when n_col is specified and vertex sizes are inended to be scaled. If this is not specified, then the function attempts to select appropriate values based on size of the figure being generated.

vertex.label.cex
(optional) Size of vertex labels, passed to igraph::plot.igraph. As with vertex.size, by default, the value is NULL, and the function attempts to pick a reasonable value based on the canvas size and the number of gene sets.

vertex.label.col
(optional) Color of vertex labels, passed to igraph::plot.igraph. If not specified, the function attempts to pick a contrasting color for vertex label text using the contrasting_color.fun argument. (default: NULL)

vertex.frame.color
(optional) Color of the vertex border. (default par('fg'))

contrasting_color.fun
(optional) A function to pick a color for vertex labels that contrasts with the vertex fill color. If unspecified, the function attempts to pick a suitable function for generating suitable set of contrasting colors, based on the contrasting_color() function. (default: For single channel plots using color scales defined with vertex_colors, or dual channel color scales defined with vertex_colors.1, or vertex_colors.2 using yellow or orange, contrasting_color(type="binary") is used, and otherwise contrasting_color(type="blackyellow") is used.)

scale_labels_by_vertex
(optional) Logical that tells the function to scale the text in vertex labels by the size of the vertex. (default: TRUE)

max_edge_width (optional) Size of vertex labels, passed to igraph::plot.igraph. By default, the value is NULL, and the function attempts to pick a reasonable value based on the canvas size and the number of gene sets.

scale.edges.by.distance
(optional) A logical telling the function to scale edges between vertices on the basis of distance. NOTE: If optimal_extreme == "max", then smaller numbers are treated as more distant, and conversely if optimal_extreme == "min", larger numbers are treated as more distant. (default: FALSE)

color.edges.by.distance
(optional) A logical telling the function to color edges between vertices on the basis of distance. This functionality will likely be deprecated. (default: FALSE)

edge_arrow_size
(optional) Size of vertex labels, passed to igraph::plot.igraph. By default, the value is NULL, and the function attempts to pick a reasonable value based on the canvas size and the number of gene sets.

seed (optional) This is a seed that the function uses to generate a plot layout. By default it is 29189892, and this results in a repeatable behavior for plots. However, to randomize the plot layout behavior, this value may be set to NULL, or if some other repeatable layout is desired, another seed may be used.

layout (optional) Either a function that generates a layout or a numerical matrix containing a vertex layout with two columns corresponding to x and y coordinates.

This argument is passed to the igraph plot method that is subsequently called by gsnPlotNetwork.old() (see .plot, below). The default layout is the anonymous function function(x){igraph::layout_with_fr(x, grid = "nogrid")}, which calls igraph::layout_with_fr() (implementing Fruchterman-Reingold layout) with the grid="nogrid" option, enabling proper layout of networks with >= 1000 gene set vertices. Other useful layouts for igraph networks include igraph::layout_with_fr (default Fruchterman-Reingold), igraph::layout_with_dh (implementing Davidson-Harel layout), igraph::layout_as_tree, igraph::layout_nicely, and others. For more details about layouts, see igraph.plotting.

.plot (optional) A plot function used to render the internally generated igraph object. By default igraph::plot.igraph is used, but for interactive plotting, igraph::tkplot may be used. For more details about plotting, see igraph.plotting.

show.legend (optional) A logical value telling the function whether or not to show legends. Legends for vertex size and node color are currently supported. (default: TRUE)

legend.lab.cex (optional) The font size of legend label text as cex units. If not specified, the function will attempt to pick an appropriate value based on the figure layout.

legend.axis.cex (optional) The font size of legend axis text as cex units. If not specified, the function will attempt to pick an appropriate value based on the figure layout.

legend.fg (optional) The foreground color of the legend that controls the color of text, axes, axis labels, ticks, and legend border. (default: par('fg'))

legend.bg (optional) The background color of the legend. This argument doesn't currently work, and may be removed in the future. (default: "#CCCCCC")

legend.vertex.fg (optional) The border color of vertices for vertex size legends. This argument allows the legend vertex frame color to be set separately from vertex.frame.color. (default: vertex.frame.color)

legend.vertex.bg (optional) The fill color of vertices for vertex size legends. (default: "#DDDDDD")

font_face (optional) The font face used for the figure. (default: par("family"))

main (optional) The plot title. (default: NULL)

cex.main (optional) The font size in cex units of the main title. (default: 1.5 * par('cex'))

mar.main (optional) The number of lines set aside for a main title when main is used. (default: 3.2)

lines.main (optional) The distance of the main title in lines from the top of the plot. (default: 0.9)

.mar.plot (optional) The margins of the plot itself. If unnspecified, the function will attempt to reserve enough room to the right of the plot for the legend or legends.

draw.legend.box.bool (option) Logical indicating whether bounding boxes should be drawn for the legends.

legend.free.cex.bool (optional) Logical allowing independent optimized sizing of legend label font sizes if TRUE. (default: FALSE)

legend_x_size.in (optional) The width of the legend in inches. If not set, the function attempts to choose an appropriate value. (default: min(2,max(width*2/5,width-height)))

colors.n (optional) The number of colors in for each channel in 1 or 2 channel plots. For single channel plots the number of colors is simply equal to this number. For dual channel plots the total number of colors in the legend is equal to the square of this number. (default: 100)

new (optional) Logical telling the function (if true) that a new plot should be added to an existing device (if TRUE) or that the current device should be cleared and written over (if FALSE). (default: FALSE)

legend_spacing.x.in (optional) Space between plot and legend in inches. This can be used to adjust the horizontal position and move the legend closer to or farther away from the plot region. Since the netork plot may not fill the entire plotting region, it may be useful to use negative values to move the legends closer to the plot. (default: 2 character widths)

legend_spacing.y.in (optional) Space between legends in inches. (default: 1 character height)

DO_BROWSER (option) Logical indicating whether browser() should be run for this function. (For debugging purposes, will probably remove.)

Details

This function is primarily for taking GSNData object containing a distance matrix, an associated edges edge-list and pathways data, and generating and rendering a corresponding igraph object. The function attempts to plot the corresponding network with vertices labeled with a gene set ID and corresponding Title, and colored according to the significance values represented in stat_col using sig_order as an indicator of whether high or low values are more significant. Edges are scaled by the value of the value of the distance statistic in the pared distance matrix.

When the parameters vertex.shape, vertex.size, vertex.label.cex, max_edge_width, and edge_arrow_size are not specified, the function attempts to pick reasonable values. These parameters are assembled into a list and attached to the returned igraph object as an attribute named GSNA_plot_params. To optimize plots, the user can examine these parameters by calling the following on the output of the function:

```
attr( x = nw.igraph, which = "GSNA_plot_params" )
```

Value

An igraph network object is returned, invisibly.

See Also

plot.GSNData gsnToIgraph plot.igraph

Examples

```
## Not run:
gsnPlotNetwork.old( object = analysis.GSN )

## End(Not run)
```

gsnPlotNetwork.old *gsnPlotNetwork.old*

Description

Function for plotting the networks within GSNData objects.

Usage

```
gsnPlotNetwork.old(
  object,
  pathways.data = NULL,
  distance = NULL,
  id_col = NULL,
  substitute_id_col = NULL,
  stat_col = NULL,
  stat_col_2 = NULL,
  sig_order = NULL,
  sig_order_2 = NULL,
  optimal_extreme = NULL,
  transform_function = nzLog10,
  pathways_title_col = "Title",
 edge_colors = c("black", "purple", "blue", "green", "yellow4", "orange", "red"),
  vertex_colors = c("white", "yellow", "red"),
  vertex_colors.1 = c("white", "red"),
  vertex_colors.2 = c("white", "blue"),
  filename = NULL,
  out_format = NULL,
  width = NULL,
  height = NULL,
  vertex.shape = "circle",
  vertex.size = NULL,
  vertex.label.cex = NULL,
  max_edge_width = NULL,
  edge_arrow_size = NULL,
  seed = 29189892,
  layout = function(x) {
     igraph::layout_with_fr(x, grid = "nogrid")
 },
  .plot = igraph::plot.igraph
)
```

Arguments

object	A GSNData object containing a pared distance matrix with edges.
pathways.data	(optional) data.frame containing associated pathways data. This defaults to whatever pathways data has already been imported into this GSNData object in object$pathways$data.
distance	(optional) The name of a distance metric used, defaults to whatever default_distance is.

`id_col` (optional) This is the name of the column in the pathways data.frame that corresponds to the names of gene sets. The default value is specified by `object$pathways$id_col`. (See details.)

`substitute_id_col` (optional) This is the name of the column that is to be substituted for the `id_col` column when labeling network vertices. (See details.)

`stat_col` (optional) This is the name of the column in the pathways data.frame that contains a significance value for coloring network vertices. The default value is specified by `object$pathways$stat_col`.

`stat_col_2` (optional) This is the name of an optional second column in the pathways data.frame that contains a significance value for coloring network vertices in a 2-color network. The default value is specified by `object$pathways$stat_col_2`. When specified, a 2-color network is generated. To force a 2-color network to plot as a standard 1-color network using `stat_col` alone, use `stat_col_2 = NA`.

`sig_order` (optional) This indicates the behavior of `stat_col`, whether low values (`'loToHi'`) or high values (`'hiToLo'`) are most significant. The default value is specified in `object$pathways$sig_order`.

`sig_order_2` (optional) This indicates the behavior of `stat_col_2`, whether low values (`'loToHi'`) or high values (`'hiToLo'`) are most significant. The default value is specified in `object$pathways$sig_order_2`.

`optimal_extreme` (optional) This indicates the behavior of the statistic used to generate the distance metric, specifically whether low values (`'min'`) or high values `'max'` are to be regarded as close. This is used for scaling the width and the color of the edges connecting vertices.

`transform_function` (optional) This is a function to transform the values in `stat_col` so that they are amenable to color-scaling. For *p*-values, a log transformation is often useful, but can produce negative infinities if the transformation is applied to zero. By default the function is the `nzLog10` (non-zero log10) function, provided by this package, which adds a small pseudocount to p-values when log10 transforming values equal to zero. If values in `stat_col` are less than zero, then log10 transformation is inappropriate and will introduce NAs, and therefore some other method should be used.

`pathways_title_col` (optional) Indicates a column to be used as the 'Title' column for network vertices. (See details.)

`edge_colors` (optional) A vector of colors included to generate a scale represent the numerical value of the edge distances. By default, the colors are arranged as a rainbow with black and purple representing the greatest distannces, and orange and red the nearest distances.

`vertex_colors` (optional) This is the standard set of colors used for a standard single color network. By default, c("white","yellow","red") is used, coloring low values white, high values red, and intermediate values yellow if `sig_order` is "loToHi" and vice versa if sig_order is "hiToLo".

`vertex_colors.1` (optional) This is the range of colors used for a 2-color network corresponding to values of `stat_col`. Up to 2 colors can be used, and should correspond to a color contrasting with `vertex_colors.2`. The default is c("white","red"), coloring high values red and low values white if `sig_order` is `"loToHi"` and vice versa if `sig_order` is `"hiToLo"`.

vertex_colors.2
: (optional) This is the range of colors used for a 2-color network corresponding to values of stat_col_2. Up to 2 colors can be used, and should correspond to a color contrasting with vertex_colors.2. The default is c("white","blue"), coloring high values blue and low values white if sig_order_2 is "loToHi" and vice versa if sig_order is "hiToLo".

filename
: (optional) An output file name for the plot. If 'out_format' is not set (see below), the output file type will be determined by the file suffix, which can be '.svg', '.pdf', or '.png'. If the out_format cannot be determined from the file name, than it may be manually set with out_format. If the output file type cannot be determined from the filename or out_format arguments, an error will be thrown.

out_format
: (optional) Output filetype when filename is specified.

width
: (optional) Sets the width of the output canvas in inches. Defaults to the width of the present graphical device.

height
: (optiona) Sets the height of the output canvas in inches. Defailts to the height of the present graphical device.

vertex.shape
: (optional) Shape of the vertex, passed to igraph::plot.igraph. By default, the value is 'circle'.

vertex.size
: (optional) Size of vertices, passed to igraph::plot.igraph. By default, the value is NULL, and the function attempts to pick a reasonable value based on the canvas size and the number of gene sets.

vertex.label.cex
: (optional) Size of vertex labels, passed to igraph::plot.igraph. As with vertex.size, by default, the value is NULL, and the function attempts to pick a reasonable value based on the canvas size and the number of gene sets.

max_edge_width
: (optional) Size of vertex labels, passed to igraph::plot.igraph. By default, the value is NULL, and the function attempts to pick a reasonable value based on the canvas size and the number of gene sets.

edge_arrow_size
: (optional) Size of vertex labels, passed to igraph::plot.igraph. By default, the value is NULL, and the function attempts to pick a reasonable value based on the canvas size and the number of gene sets.

seed
: (optional) This is a seed that the function uses to generate a plot layout. By default it is 29189892, and this results in a repeatable behavior for plots. However, to randomize the plot layout behavior, this value may be set to NULL, or if some other repeatable layout is desired, another seed may be used.

layout
: (optional) Either a function that generates a layout or a numerical matrix containing a vertex layout with two columns corresponding to *x* and *y* coordinates. This argument is passed to the igraph plot method that is subsequently called by gsnPlotNetwork.old() (see .plot, below). The default layout is the anonymous function function(x){igraph::layout_with_fr(x, grid = "nogrid")}, which calls igraph::layout_with_fr() (implementing Fruchterman-Reingold layout) with the grid="nogrid" option, enabling proper layout of networks with >= 1000 gene set vertices. Other useful layouts for igraph networks include igraph::layout_with_fr (default Fruchterman-Reingold), igraph::layout_with_dh (implementing Davidson-Harel layout), igraph::layout_as_tree, igraph::layout_nicely, and others. For more details about layouts, see igraph.plotting.

.plot
: (optional) A plot function used to render the internally generated igraph object. By default igraph::plot.igraph is used, but for interactive plotting, igraph::tkplot may be used. For more details about plotting, see igraph.plotting.

Details

This function is primarily for taking GSNData object containing a distance matrix, an associated edges edge-list and pathways data, and generating and rendering a corresponding igraph object. The function attempts to plot the corresponding network with vertices labeled with a gene set ID and corresponding Title, and colored according to the significance values represented in stat_col using sig_order as an indicator of whether high or low values are more significant. Edges are scaled by the value of the value of the distance statistic in the pared distance matrix.

When the parameters vertex.shape, vertex.size, vertex.label.cex, max_edge_width, and edge_arrow_size are not specified, the function attempts to pick reasonable values. These parameters are assembled into a list and attached to the returned igraph object as an attribute named GSNA_plot_params. To optimize plots, the user can examine these parameters by calling the following on the output of the function:

```
attr( x = nw.igraph, which = "GSNA_plot_params" )
```

Value

An igraph network object is returned, invisibly.

See Also

plot.GSNData gsnToIgraph plot.igraph

Examples

```
## Not run:
gsnPlotNetwork.old( object = analysis.GSN )

## End(Not run)
```

---

gsnSubnetSummary *gsnSubnetSummary*

Description

Generates a table summarizing subnets that incorporates subnets and pathways data.

Usage

```
gsnSubnetSummary(
  object,
  pathways.data = NULL,
  distance = NULL,
  id_col = NULL,
  stat_col = NULL,
  sig_order = NULL,
  stat_col_2 = NULL,
  sig_order_2 = NULL,
  summary_statistics = c("hm", "min_max"),
  seed_gs_fields = NULL
)
```

Arguments

object	A GSNData data object containing a distance matrix and subnets data. If pathways data is not specified by the pathways.data argument (described below), the object must contain imported pathways data as well.
pathways.data	An (optional) data.frame containing pathways data (GSEA, CERNO, GSNORA, etc.) with 1 or 2 associated statistical columns, typically *P*-values, specified by stat_col and stat_col_2 below.
distance	A distance metric with associated subnets data.
id_col	(optional) This is the name of the column in the pathways data.frame that corresponds to the names of gene sets. The default value is specified by `object$pathways$id_col`. (See details.)
stat_col	(optional) Specifies the name of the first statistical column, if not specified, defefaults to the value in `object$pathways$stat_col`.
sig_order	(optional) This indicates the behavior of `stat_col`, whether low values (`'loToHi'`) or high values (`'hiToLo'`) are most significant. The default value is specified in `object$pathways$sig_order`.
stat_col_2	(optional) Specifies the name of the second statistical column, if not specified, defefaults to the value in `object$pathways$stat_col_2`.
sig_order_2	(optional) This indicates the behavior of `stat_col_2`, whether low values (`'loToHi'`) or high values (`'hiToLo'`) are most significant. The default value is specified in `object$pathways$sig_order_2`.
summary_statistics	(optional) A character vector specifying which summary statistics are to be calculated from the 'stat_col'. Acceptable values include 'hm' specifying harmonic mean, 'min_max', specifying either minimum or maximum depending on `sig_order`, or the name of a function. (default: `c('hm', 'min_max')`)
seed_gs_fields	(optional) A character vector specifying the names of additional seed gene set fields to retain from pathways data.

Details

The output data.frame contains a list of subnets, each with an associated list of gene set IDs. For each subnet, summary statistics are calculated, including the harmonic mean of `stat_col` and (if specified) `stat_col_2`. In addition, the minimum or maximum of the `stat_col` and `stat_col_2` is calculated, depending on the `sig_order` and `sig_order_2`. For `loToHi`, the minumum is calculated, and for `hiToLo`, the maximum.

Value

A data.frame with a statistical summary of subnets.

Examples

```
## Not run:
   subnetSummary.df <- gsnSubnetSummary( object = analysis.GSN )

## End(Not run)
```

gsnSubset *gsnSubset*

Description

Create a subset GSNData object, containing only a subset of vertices or subnets.

Usage

```
gsnSubset(object, distance = NULL, vertex_names = c(), subnet_names = NULL)
```

Arguments

object	A GSNData object.
distance	Specifies a distance metric to use for subsetting. Defaults to the default_distance.
vertex_names	A chatacter vector specifying the vertex names/gene sets to include in the GSNData subset object.
subnet_names	A chatacter vector specifying the names of the subnets to include in the GSNData subset object.

Details

This function is useful for subsetting a single subnet, or a small set of subnets for the purpose of plotting just that subnet.

Value

A new GSNData object is returned containing a subset of the vertices and subnets from the original GSNData object. For a given distance, the following data are subsetted for the included vertices and copied:

`$distances[[distance]]$matrix`
: The raw distance matrix, subsetted.

`$distances[[distance]]$pared`
: The pared distance matrix, subsetted.

`$distances[[distance]]$edges`
: The edge list, subsetted

`$distances[[distance]]$vertex_subnets`
: The vertex assignments for each subnet, subsetted.

`$distances[[distance]]$clusters`
: The cluster assignments for each subnet, subsetted (for hierarchical clustering).

`$distances[[distance]]$optimal_extreme`
: Character vector of length 1 indicating whether min or max distances are close in the raw distance matrix.

`$distances[[distance]]$pared_optimal_extreme`
: Character vector of length 1 indicating whether min or max distances are close in the pared distance matrix.

The hclust object generated by hierarchical clustering is not currently subsetted or copied.

The `default_distance` is set as whichever distance matrix is copied. Currently, this function only supports copying a single distance matrix.

The following pathwayys data are copied:

`$pathways$data` Pathways results, subsetted.

`$pathways$type` The type of pathways results, copied.

`$pathways$id_col`
: The identifier, copied.

`$pathways$stat_col`
: Statistical column name, copied.

`$pathways$sig_order`
: The significance order of the pathways results, based on the stat_column; are the pathways results to be sorted by significance from 'loToHi' (most significant values are low) or 'hiToLo' (most significant values high)?

See Also

`GSNData()`

Examples

```
## Not run:
analysis.GSN.SN1 <- gsnSubset( analysis.GSN, subnet_names=c("1") )

## End(Not run)
```

`gsnToIgraph` *gsnToIgraph*

Description

For a `GSNData` object containing an edge list, generate an igraph object.

Usage

```
gsnToIgraph(object, distance = NULL)
```

Arguments

`object`	A `GSNData` object containing a pared distance matrix and an edge list.
`distance`	(optional) A character vector specifying a distance to use. If no `distance` is specified, the value of the `default_distance` will be used.

Details

This is used by gsnPlotNetwork to generate an `igraph`. Users will probably not need to call gsnToIgraph, for most cases. If edges are not found, it will emit an error.

Value

Returns an igraph object corresponding to the edges and vertices in the GSNData object's edge-list data.frame.

See Also

gsnPlotNetwork() plot.GSNData()

Examples

```
## Not run:
   network.igraph <- gsnToIgraph( object = object.GSN, distance = 'stlf' )

## End(Not run)
```

---

gsn_default_distance *gsn_default_distance*

Description

Retrieve or set default distances in a GSNData object.

Usage

```
gsn_default_distance(object)

gsn_default_distance(object) <- value
```

Arguments

object	An GSNData object.
value	A character vector of length 1 containing the name of a valid distance metric. The value must be a valid distance metric, for which there exists a distance matrix in the GSNData object, or else an error will be thrown.

Value

The name of the default distance metric.

See Also

gsn_distances

Examples

```
## Not run:
  # Print the value of the default_distance:
  gsn_default_distance( analysis.GSN )

## End(Not run)
## Not run:
# Set the value of the default_distance to 'jaccard':
gsn_default_distance( analysis.GSN ) <- 'jaccard'

## End(Not run)
```

---

gsn_distances *gsn_distances*

Description

Given a GSNData object, returns a character vector of distance matrices that are contained within.

Usage

```
gsn_distances(object)
```

Arguments

object	An object of type GSNData.

Value

A character vector containing the names of distance matrices.

See Also

gsn_default_distance()

Examples

```
## Not run:
# Print the names of distances in the GSNData object:
gsn_distances( analysis.GSN )

## End(Not run)
```

intV2Color *intV2Color*

Description

Converts a numeric or integer vector of length 3 containing RGB values in the range of 0 to 255 to 24 bit color specifications in the form "#FFFFFF".

Usage

```
intV2Color(rgb_v)
```

Arguments

rgb_v	An integer or numeric vector of length 3 containing RGB channel intensities from 0 to 255.

Value

A 24-bit color specification in the form "#FFFFFF".

See Also

color2IntV()

Examples

```
col_v <- c( 255, 100, 240)
col <- intV2Color( col_v )
```

lfisher_cpp *lfisher_cpp*

Description

Takes a four integers corresponding to a 2x2 contingency matrix and calculates a natural-log transformed Fisher *p*-value.

Usage

```
lfisher_cpp(a, b, c, d, e_precision = 12, alternative = 1L)
```

Arguments

a	(required) This corresponds to cell 1 of the contingency matrix. In the context of GSNA, assuming two gene sets, this is used as the number of observable genes in the background that are not present either gene set.
b	(required) This corresponds to cell 2 of the contingency matrix. For GSNA this is the number of observable genes present in gene set 1, but not gene set 2.
c	(required) This corresponds to cell 3 of the contingency matrix. For GSNA this is the number of observable genes present in gene set 2, but not gene set 1.
d	(required) This corresponds to cell 4 of the contingency matrix. For GSNA this is the number of observable genes present in both gene sets 1 and 2.
e_precision	(optional) Numeric value that determines the precision of summation of partial *p*-values. For the less-than, greater-than and two-sided options in calculating the log-Fisher *p*-value, log-space summation of partial *p*-values must be accomplished using the so-called *Log-Sum-Exponent* trick. Due to limitations of precision in C++, however, numbers that differ by more than about 11 powers of *e* cannot be summed. By specifying a value less than 12, slightly less precise p-values can be calculated slightly faster. This option was included as way to accelerate calculation of *p*-values, but has not proven to significantly improve performance, so it may be removed in the future. Defaults to 12.
alternative	(optional) Integer corresponding to 4 options: 1 single-sided, greater-than. Sums *p*-values for intersections greater than and equal to d. 2 single-sided, less-than. Sums *p*-values for intersections less than and equal to d. 3 two-sided, sums all partial *p*-values less than or equal to the partial *p*-value for intersections equal to d. 4 partial. Calculates single *p*-value for intersections equal to d.

Details

Calculation of Fisher *p*-values is discussed in detail elsewhere, but partial natural-log transformed *p*-values are calculated as follows:

Given a 2x2 contingency matrix of the form:

```
| a b |
| c d |
```

The natural log of the partial *p*-values is given by:

```
log.p  = ln((a+b)!) + ln((c+d)!) + ln((a+c)!) +  ln((b+d)!) -
      ln(a!) - ln(b!) - ln(c!) - ln(d!) - ln( (a + b + c + d)!)
```

For the single and two-tailed alternatives, partial *p*-values are summed using the so-called 'log-sum-exponent' method.

Value

This function returns a numeric (double in C++) natural log-Fisher p-value.

See Also

gsIntersectCounts scoreLFMatrix_C

Examples

```
## Not run:
library( GSNA )
log_fisher_p <- lfisher_cpp( a = 16000, b = 200, c = 170, d = 100, alternative = 3 )

## End(Not run)
```

---

make1ColorLegend *make1ColorLegend*

Description

make1ColorLegend

Usage

```
make1ColorLegend(
  numbers,
  oneColorEncode.fun,
  n = 100,
  lab = NULL,
  log_scale = FALSE,
  .plt.leg = c(0.71, 1, 0.7, 1),
  .mar.leg.vm = adj_mar_leg_vm(.mar.leg.vm = c(1.1, 4.1, 1.1, 1.1)),
  .fin = graphics::par("fin"),
  h_w.leg = 1,
  legend_thickness = NULL,
  legend.lab = "",
  cex.lab = NULL,
  cex.axis = NULL,
  axis_lab_ratio = 0.9,
  legend.fg = graphics::par("fg"),
  legend.bg = graphics::par("bg"),
  draw.legend.box.bool = FALSE,
  v.adjust = "top",
  h.adjust = "center",
  render.bool = TRUE,
  restore.params.bool = TRUE,
  optimize.legend.size = FALSE
)
```

Arguments

numbers	Numbers to set the range of colors.
oneColorEncode.fun	A function that takes a numeric value and returns an encoded RGB color value.

n	(optional) The number of color gradations to include in the legend (default 100).
lab	(optional) The axis label. Since the color scale is vertical, this is a y-axis label.(default: NULL)
log_scale	Boolean value indicating whether the scale should be log scale. (default: FALSE)
.plt.leg	A vector of 4 coordinates indicating the region where the legend is to be ploted.
.mar.leg.vm	A vector of 4 coordinates indicating legend margins. These values are picked automatically depending on the available geometry, so in general, you won't want to change this.
.fin	(optional) Figure width and height of the figure in inches (defaults to par('fin')).
h_w.leg	(optional) Height to width ratio of the 1-color legend panel. (default: 1)
legend_thickness	(optional) The width of the legend in raster matrix cells. (default: approximately 15% of height)
legend.lab	(optional) The legend label. (default '')
cex.lab	(optional) Font cex size for labels. If unspecified, then the function will attempt to pick an appropriate value.
cex.axis	(optional) Font cex size for axes. If unspecified, then the function will attempt to pick an appropriate value.
axis_lab_ratio	(optional) If cex.lab and cex.axis are unspecified, the function will attempt to pick appropriate values. This argument is the ratio of axis marks to axis labels (default 0.9).
legend.fg	(optional) The forground color of the legend (by default inherited from par('fg')).
legend.bg	(optional) The background color of the legend (by default inherited from par('bg')).
draw.legend.box.bool	(optional) Boolean indicated whether a box should be drawn around the legend. (default: FALSE)
v.adjust	(optional) When the size of the legend is optimized for the available space, indicates whether the legend should be adjusted towards the top, bottom, or middle of the available space. (default: 'top')
h.adjust	(optional) When the size of the legend is optimized for the available space, indicates whether the legend should be adjusted towards the left, right or center of the available space. (default" 'center')
render.bool	(optional) Boolean indicating whether the legend should be rendered, or just return graphical parameters. (default: TRUE)
restore.params.bool	(optional) Boolean indicating whether graphical parameters should be restored to original values once the legend is drawn. (default: TRUE)
optimize.legend.size	(optional) Boolean indicated whether the function should attempt to optimize the size of the legend. (default: FALSE)

Value

Invisible list of graphical parameters.

make2ColorLegend *make2ColorLegend*

Description

This function generates a 2-color legend for network plots and dendrograms, consisting of a square raster with x&y scales and labels.

Usage

```
make2ColorLegend(
  numbers.1,
  numbers.2,
  twoColorEncode.fun,
  n = 100,
  lab.1 = NULL,
  lab.2 = NULL,
  cex.lab = NULL,
  cex.axis = NULL,
  axis_lab_ratio = 0.9,
  legend.scale.factor = 1.15,
  legend.ylab = NULL,
  legend.xlab = NULL,
  legend.fg = graphics::par("fg"),
  legend.bg = graphics::par("bg"),
  log_scale.1 = FALSE,
  log_scale.2 = FALSE,
  .plt.leg = c(0.71, 1, 0.7, 1),
  .mar.leg.vm = adj_mar_leg_vm(.mar.leg.vm = c(4.1, 4.1, 2.1, 2.1)),
  .fin = graphics::par("fin"),
  v.adjust = "top",
  h.adjust = "center",
  draw.legend.box.bool = FALSE,
  optimize.legend.size = FALSE,
  render.bool = TRUE,
  restore.params.bool = TRUE
)
```

Arguments

numbers.1	Numbers to set the range of colors for the y-axis.
numbers.2	Numbers to set the range of colors for the x-axis.
twoColorEncode.fun	A function that takes two numeric values and returns an encoded RGB color value.
n	(optional) The number of color gradations per channel to include in the legend (default 100)
lab.1	(optional) y-axis label (default: NULL)
lab.2	(optional) x-axis label (default: NULL)

`cex.lab` (optional) Font cex size for labels. If unspecified, then the function will attempt to pick an appropriate value.

`cex.axis` (optional) Font cex size for axes. If unspecified, then the function will attempt to pick an appropriate value.

`axis_lab_ratio` (optional) If cex.lab and cex.axis are unspecified, the function will attempt to pick appropriate values. This argument is the ratio of axis marks to axis labels (default 0.9).

`legend.scale.factor` (optional) A fudge factor for scaling the legend (default 1.15).

`legend.ylab` (optional) The y label of the legend.

`legend.xlab` (optional) The x label of the legend.

`legend.fg` (optional) The forground color of the legend (by default inherited from `graphics::par('fg')`).

`legend.bg` (optional) The background color of the legend (by default inherited from `graphics::par('bg')`).

`log_scale.1` (optional) Indicates whether the y-values are log scale or not. (default: FALSE)

`log_scale.2` (optional) Indicates whether the x-values are log scale or not. (default: FALSE)

`.plt.leg` A vector of 4 coordinates indicating the region where the legend is to be ploted.

`.mar.leg.vm` (optional) A vector of 4 coordinates indicating legend margins. These values are picked automatically depending on the available geometry, so in general, you won't want to change this.

`.fin` (optional) Figure width and height of the figure in inches (defaults to `graphics::par('fin')`).

`v.adjust` (optional) When the size of the legend is optimized for the available space, indicates whether the legend should be adjusted towards the top, bottom, or middle of the available space. (default: `'top'`)

`h.adjust` (optional) When the size of the legend is optimized for the available space, indicates whether the legend should be adjusted towards the left, right or center of the available space. (default" `'center'`)

`draw.legend.box.bool` (optional) Boolean indicated whether a box should be drawn around the legend. (default: `FALSE`)

`optimize.legend.size` (optional) Boolean indicated whether the function should attempt to optimize the size of the legend. (default: `FALSE`)

`render.bool` (optional) Boolean indicating whether the legend should be rendered, or just return graphical parameters. (default: `TRUE`)

`restore.params.bool` (optional) Boolean indicating whether graphical parameters should be restored to original values once the legend is drawn. (default: `TRUE`)

Value

Invisible list of graphical parameters.

make2ColorLegendStack *make2ColorLegendStack*

Description

make2ColorLegendStack

Usage

```
make2ColorLegendStack(numbers.1, numbers.2, twoColorEncode.fun, n = 100)
```

Arguments

numbers.1	A set of numbers to define the range of channel 1 numerical values to be represented in the legend. Only the extreme min and max values are necessary.
numbers.2	A set of numbers to define the range of channel 2 numerical values to be represented in the legend. Only the extreme min and max values are necessary.
twoColorEncode.fun	A function to take two sets of numbers and return a matrix of colors
n	The number of of color gradations per channel to render. The total number of gradations to render is the square of this number.

Details

Generates a raster stack for a 2 color legend.

Value

A raster stack for a 2 color legend.

makeFilteredGenePresenceAbsenceMatrix *makeFilteredGenePresenceAbsenceMatrix*

Description

Take character vector containing the set of observable genes in a data set and a gene set collection and generate a presence/absence matrix of observable genes in each gene set/module.

Usage

```
makeFilteredGenePresenceAbsenceMatrix(ref.background, geneSetCollection)
```

Arguments

ref.background	(required) A character vector corresponding to the genes observable in a differential expression, ATACSeq or other dataset. This corresponds to the background used in tools like DAVID.
geneSetCollection	(required) A gene set collection either in the form of a tmod object, or a list of gene sets / modules as character vectors containing gene symbols and names corresponding to the gene module identifier.

Value

This returns a gene presence/absence matrix with genes corresponding to rows, gene sets/modules corresponding to columns, and TRUE or FALSE values corresponding to presence or absence of a particular gene in a particular gene set/module. This matrix has been filtered to only enclude genes observable in a data set.

See Also

buildGeneSetNetworkLFFast buildGeneSetNetworkSTLF buildGeneSetNetworkJaccard

Examples

```
## Not run:
library(GSNA)
observable_genes <- rownames(FILTERED_RNASEQ_COUNT_MATRIX)
msig.subset <- msig[cerno_results$ID,]
filteredGenePresenceAbsence_Matrix <-
       makeFilteredGenePresenceAbsenceMatrix( ref.background = observable_genes,
                                              geneSetCollection = msig.subset )

## End(Not run)
```

---

makeLeafSizeLegend *makeLeafSizeLegend*

Description

Internal GSNA package function to generate a leaf size legend for GSNA network plots generated via gsnHierarchicalDendrogram().

Usage

```
makeLeafSizeLegend(
  numbers,
  sizeEncode.fun,
  pch = 16,
  cin.pch = c(0.125, 0.125),
  .plt.leg,
  log_scale = NULL,
  cex.ticks = graphics::par("cex"),
  leaf.col = "#999999",
  leaf_border_color = "#666666",
  legend.lab = NULL,
  legend.lab.cex = NULL,
  legend.fg = graphics::par("fg"),
  legend.bg = graphics::par("bg"),
  font_face = graphics::par("family"),
  .fin = graphics::par("fin"),
  order_high_to_low = FALSE,
  optimize.legend.size = FALSE,
  y.compression.factor = 1,
```

```
  v.adjust = "top",
  h.adjust = "left",
  draw.legend.box.bool = FALSE,
  bottom_legend_margin = 0.25,
  restore.params.bool = TRUE,
  render.bool = TRUE
)
```

Arguments

numbers	A vector containing numerical values to be mapped to a range of leaf sizes. The only really needs to be a mininimum and a maximum value to establish a set of scale values.
sizeEncode.fun	The function used by gsnPlotNetwork() to convert the value in n_col (usually representing gene set sizes) into leaf sizes in cex units.
pch	(optional) A pch symbol used for representing leaves in the legend, generally the same as is used in the dendrogram itself. (default: 16 (a closed circle))
cin.pch	(optional) A numeric vector of length 2 representing the width and height in inches of the pch used in the dendrogram when cex = 1. This is used for calculating the height and width of lines of the legend, since not all pch characters are the same size and they tend to be smaller than the corresponding character sizes in inches. The default should be a good approximation for most pch characters. (default: c(0.125,0.125))
.plt.leg	Required plot area where the legend is drawn, specified in the manner of graphics::par('plt') as a vector of four values in figure units. This is generally determined before rendering by calling makeNodeSizeLegend() and the other legend plot functions with a provisional value for .plt.leg that specifies the maximal available region for plotting the legend and the arguments render.bool = FALSE, optimize.legend.size = TRUE and h.adjust specified as "left", "right", or "center" prior to rendering. The function returns a list of graphical parameters including an optimized .plt.leg (see value) and the different values for this returned by the various legend plot functions can be reconciled prior to calling this function a second time with render.bool = TRUE to actually render the legend.
log_scale	(optional) Logical value indicating whether the size values should be incremented in a linear or logrithmic scale. If not specified, then this will be decided based on the range of minimum to maximum values specified in the numbers argument.
cex.ticks	(optional) The font size used for tick labels. (default: graphics::par('cex'))
leaf.col	(optional) The color of the leaf symbols used in the legend, or for open symbols, pch c(21, 22, 23, 24, 25), the background fill color. (default: "#999999")
leaf_border_color	(optional) For open symbols, pch c(21, 22, 23, 24, 25), the color of the symbol border. (default: "#666666")
legend.lab	(optional) A title for the legend. (default: NULL)
legend.lab.cex	(optional) The font size of the legend labels in cex units. (default: cex.ticks * 1.1)
legend.fg	(optional) Legend foreground, used for text and border color. (default: graphics::par('fg'))

`legend.bg` (optional) Legend background. Doesn't currently do anything. (default: graphics::par('bg'))

`font_face` (optional) The font family used for text in the legend. (default: graphics::par("family"))

`.fin` (optional) (optional) The width and height of the current figure in inches. It is advisable to allow this to be automatically determined. (default: graphics::par('fin'))

`order_high_to_low` (optional) If TRUE, tells the function to order the vertices in the legend with the largest first, and the smallest last. Otherwise, vertices are ordered from lowest to highest. (default FALSE)

`optimize.legend.size` (optional) Tells the function to optimize the legend dimensions and graphical parameters. (default: FALSE)

`y.compression.factor` (optional) A compression factor allowing lines of the legend to be squeezed together if less than one or stretched apart if greater than one. When less than 1, this can result in vertices overlapping. (default: 1)

`v.adjust` (optional) This argument, which is passed to the function `adjust_plt()` controls how the legend position and height are optimized. Valid values are "top", "center", "bottom" and NULL. See `adjust_plt()` for more details. (default: "top")

`h.adjust` (optional) This argument, which is passed to the function `adjust_plt()` controls how the legend position and width are optimized. Valid values are "left", "center", "right" and NULL. See `adjust_plt()` for more details.

`draw.legend.box.bool` (optional) This argument is a logical, telling the function to draw a box around the legend. (default: FALSE)

`bottom_legend_margin` (optional) Number of lines added to the bottom to create a bottom legend margin. (default: FALSE)

`restore.params.bool` (optional) Logical telling the function to restore any graphical parameters that may have been changed to their values when the function was called. (restore.params.bool: TRUE)

`render.bool` (optional) Logical telling the function to actually render the legend as opposed to merely calculating graphical parameters. (default: TRUE)

Value

The function returns a list with a set of graphic parameters, including the optimized value `.plt.leg` if `optimize.legend.size = TRUE`.

Examples

```
## Not run:
    legend.dat <- makeLeafSizeLegend( numbers = gs_numbers,
             sizeEncode.fun = sizeEncode.fun,
             .plt.leg = c( legend_left_x.fig,
                           legend_left_x.fig + legend_x_size.fig,
                           0,
```

```
                    legend_top_y.fig
  ),
  legend.lab = n_col,
  pch = leaves_pch,
  cex = cex,
  leaf_border_color = leaf_border_color,
  leaf.col = legend.leaf.col,
  legend.fg = legend.fg,
  legend.bg = legend.bg,
  h.adjust = 'left',
  v.adjust = 'top',
  render.bool = FALSE,
  optimize.legend.size = TRUE )

## End(Not run)
```

`makeLinearNColorGradientFunction`
makeLinearNColorGradientFunction

Description

Given a set of colors and a range of values, generate a function to encode numbers in the specified range of colors. This serves as a backend for `makeOneColorEncodeFunction()` `makeTwoColorEncodeFunction()` allowing more than 2 color intervals can be specified, so that color encodings consisting of 3 or more colors per color-dimension/channel can be created.

Usage

```
makeLinearNColorGradientFunction(
  colors = c("#000000", "#CCCC00", "#FF0000"),
  x.min = 0,
  x.max = 100
)
```

Arguments

`colors`	A vector of colors, either by name or as hexadecimal colors.
`x.min`	The minimal value for the range of numbers to be encoded.
`x.max`	The maximal value for the range of numbers to be encoded.

Details

Given n colors, where n >=1 and a range of numbers from x.min to x.max, the function breaks down the range of numbers into n-1 ranges, and then maps numerical values linearly to numbers in each range bounded by successive colors. This is used by `makeOneColorEncodeFunction()` and `makeTwoColorEncodeFunction()`.

Value

Returns a function encoding a single numerical value as a numerical vector of length 3 containing RGB balues from 0 to 255.

See Also

makeOneColorEncodeFunction, makeTwoColorEncodeFunction.

Examples

```
three_col_fun <- makeLinearNColorGradientFunction( colors = c("blue", "white", "red"),
                                                   x.min = 0,
                                                    x.max = 100 )

three_col_mat <- t( sapply( c(0, 25, 50, 75, 100 ) ,three_col_fun ) )
```

---

makeNodeSizeLegend *makeNodeSizeLegend*

Description

Internal GSNA package function to generate a vertex size legend for GSNA network plots generated via gsnPlotNetwork() which uses plot.igraph() from the igraph package to generate plots.

Usage

```
makeNodeSizeLegend(
  numbers,
  sizeEncode.fun,
  usr_x_coords_per_inch = NULL,
  log_scale = NULL,
  cex.ticks = par("cex"),
  legend.lab = NULL,
  legend.lab.cex = NULL,
  legend.fg = par("fg"),
  legend.bg = par("bg"),
  legend.vertex.fg = par("fg"),
  legend.vertex.bg = "#DDDDDD",
  font_face = par("family"),
  .plt.leg,
  .fin = par("fin"),
  .pin = par("pin"),
  .usr = par("usr"),
  order_high_to_low = FALSE,
  optimize.legend.size = FALSE,
  y.compression.factor = 1,
  v.adjust = "top",
  h.adjust = "center",
  draw.legend.box.bool = FALSE,
  bottom_legend_margin = 0.25,
  restore.params.bool = TRUE,
  render.bool = TRUE
)
```

Arguments

numbers	A vector containing numerical values to be mapped to a range of vertex sizes. The only really needs to be a mininimum and a maximum value to establish a set of scale values.
sizeEncode.fun	The function used by gsnPlotNetwork() to convert the value in n_col (usually representing gene set sizes) into vertex sizes within the igraph::plot.igraph() function. For the current implementation of the igraph package, vertex sizes are user x coordinates * 200.
usr_x_coords_per_inch	The ratio of horizontal (x) user coordinates per inch at the time when (or immediately after) plot.igraph() is called, equal to (par('usr')[2] - par('usr')[1]) / par('pin')[1]. This ratio is essential for correctly sizing the vertices in the legend. If not specified when the function is called, get_usr_x_coords_per_inch will be called to obtain this value, but this is only valid if plot.window() has not been called to create a new user coordinate system.
log_scale	(optional) Logical value indicating whether the size values should be incremented in a linear or logrithmic scale. If not specified, then this will be decided based on the range of minimum to maximum values specified in the numbers argument.
cex.ticks	(optional) The font size used for tick labels. (default: par('cex'))
legend.lab	(optional) The label for the legend, generally the name of the pathways data field used for scaling the vertex sizes.
legend.lab.cex	(optional) The font size of the legend labels in cex units. (default: cex.ticks * 1.1)
legend.fg	(optional) The foreground color of the legend, used for font, tick and border color. (default: par("fg"))
legend.bg	(optional) The background color of the legend. Doesn't currently do anything. (default: par("bg"))
legend.vertex.fg	(optional) The foreground color of the legend vertices, used to set the border color of vertices in the legend. This should generally be the same as the value of vertex.frame.color assigned when generating the igraph network. (default: par("fg"))
legend.vertex.bg	(optional) The background color of the legend vertices, used to set the fill color of vertices in the legend. (default: "#DDDDDD")
font_face	(optional) The font family used for text in the legend. (default: par("family"))
.plt.leg	Required plot area where the legend is drawn, specified in the manner of par('plt') as a vector of four values in figure units. This is generally determined before rendering by calling makeNodeSizeLegend() and the other legend plot functions with a provisional value for .plt.leg that specifies the maximal available region for plotting the legend and the arguments render.bool = FALSE, optimize.legend.size = TRUE and h.adjust specified as "left", "right", or "center" prior to rendering. The function returns a list of graphical parameters including an optimized .plt.leg (see value) and the different values for this returned by the various legend plot functions can be reconciled prior to calling this function a second time with render.bool = TRUE to actually render the legend.

`.fin` (optional) The width and height of the current figure in inches. It is advisable to allow this to be automatically determined. (default: par('fin'))

`.pin` (optional) The width and height of the current plot region in inches. It is advisable to allow this to be automatically determined. (default: par('pin'))

`.usr` (optional) The range of user coordinates corresponding to the plot region. It is advisable to allow this to be automatically determined. (default: par('usr'))

`order_high_to_low` (optional) If TRUE, tells the function to order the vertices in the legend with the largest first, and the smallest last. Otherwise, vertices are ordered from lowest to highest. (default FALSE)

`optimize.legend.size` (optional) Tells the function to optimize the legend dimensions and graphical parameters. (default: FALSE)

`y.compression.factor` (optional) A compression factor allowing lines of the legend to be squeezed together if less than one or stretched apart if greater than one. When less than 1, this can result in vertices overlapping. (default: 1)

`v.adjust` (optional) This argument, which is passed to the function `adjust_plt()` controls how the legend position and height are optimized. Valid values are "top", "center", "bottom" and NULL. See `adjust_plt()` for more details. (default: "top")

`h.adjust` (optional) This argument, which is passed to the function `adjust_plt()` controls how the legend position and width are optimized. Valid values are "left", "center", "right" and NULL. See `adjust_plt()` for more details.

`draw.legend.box.bool` (optional) This argument is a logical, telling the function to draw a box around the legend. (default: FALSE)

`bottom_legend_margin` (optional) Number of lines added to the bottom to create a bottom legend margin. (default: FALSE)

`restore.params.bool` (optional) Logical telling the function to restore any graphical parameters that may have been changed to their values when the function was called. (restore.params.bool: TRUE)

`render.bool` (optional) Logical telling the function to actually render the legend as opposed to merely calculating graphical parameters. (default: TRUE)

Value

The function returns a list with a set of graphic parameters, including the optimized value `.plt.leg` if `optimize.legend.size = TRUE`.

Examples

```
## Not run:
    uxcpi <- get_usr_x_coords_per_inch()

    .plt.leg <- c( legend_left_x.fig,
                   legend_left_x.fig + legend_x_size.fig,
                   legend_bottom_y.fig
                   legend_top_y.fig )
```

```
    legend.list <- makeNodeSizeLegend( numbers = c(10, 240),
                                       sizeEncode.fun = sizeEncode.fun,
                                       .plt.leg = .plt.leg,
                                       legend.lab = "Gene Set Size",
                                       legend.lab.cex = 1,
                                       legend.fg = "black",
                                       legend.vertex.fg = "black",
                                       legend.vertex.bg = "#CCCCCC",
                                       usr_x_coords_per_inch = uxcpi,
                                       draw.legend.box.bool = TRUE,
                                       h.adjust = "left",
                                       render.bool = TRUE,
                                       optimize.legend.size = TRUE  )

## End(Not run)
```

`makeOneColorEncodeFunction` *makeOneColorEncodeFunction*

Description

Generate a function to take a numerical vector and return a color, either as a vector of hexadecimal encoded colors, or as a three column matrix.

Usage

```
makeOneColorEncodeFunction(
  numbers,
  colors = c("#FFFFFF", "#FFFF00", "#FF0000"),
  c.fun = NULL,
  na.color = "#CCCCCC"
)
```

Arguments

`numbers`	A set of numbers to define the range of numerical values for which the color encode function will be defined. Only the extreme min and max values are necessary.
`colors`	The range of colors to be returned by the function function. (default: c("#FFFFFF", "#FF0000"))
`c.fun`	(optional) A function to convert numerical values into colors. If not specified, this is generated based on `numbers` and `colors` using `makeLinearNColorGradientFunction()`.
`na.color`	(optional) The color returned from the function for NA values (default: "#CCCCCC").

Value

The fuction returns a function that takes 3 arguments and returns either a vector of hexadecimal colors or a 3-column matrix of columns. The arguments:

numbers	A vector of numbers to be encoded as a color value.
numbers.2	This argument is ignored, and is only included to be compatible with functions generated by the 2-color encoding functions that take three arguments.
output_as	Specifies the type of return value. If 'vector' or 'rgb', the function returns a vector of hexadecimal colors (e.g."#FFCCAA"), if 'matrix','array', a three column numeric matrix is returned (Columns are "R", "G", or "B"). Currently, 'vector' are synonyms 'rgb', as are 'matrix' and 'array'

.

Examples

```
# Prepare the function:
oneColorEnc.fun <- makeOneColorEncodeFunction( numbers = c( 0.4, 6 ),
                                               colors = c("white", "yellow", "red"),
                                             )

# Encode a vector of numbers as a vector of colors:
colors_as_vector <- oneColorEnc.fun( numbers = c( 0.4, 1.2, 5, 6 ),
                                     output_as = 'vector' )
```

---

makeSymmetricDist *makeSymmetricDist*

Description

Utility function to convert a matrix of non-symmetrical distances (A->B != B->A) into a symmetrical one. A method of aggregating the non-symmetrical distances can be specified. The default aggregation method is mean.

Usage

```
makeSymmetricDist(mat, FUN = mean)
```

Arguments

mat	A non-symmetrical matrix of distances.
FUN	function applied to the non-symmetrical distance pairs to aggregate into a symmetrical distance.

Value

A symmetrical distance matrix.

Examples

```
## Not run:
dist.sym <- makeSymmetricDist( mat = dist.non_sym, FUN = max )

## End(Not run)
```

`makeTwoColorEncodeFunction`
makeTwoColorEncodeFunction

Description

Generate a function to take two numerical vector arguments and return a color, either as a vector of hexadecimal encoded colors, or as a three column matrix.

Usage

```
makeTwoColorEncodeFunction(
  numbers.1,
  numbers.2,
  colors.1 = c("#FFFFFF", "#FF0000"),
  colors.2 = c("#FFFFFF", "#0000FF"),
  combine_method = "mean",
  c1.fun = NULL,
  c2.fun = NULL,
  na.color = "#CCCCCC"
)
```

Arguments

`numbers.1`	A set of numbers to define the range of channel 1 numerical values for which the color encode function will be defined. Only the extreme min and max values are necessary.
`numbers.2`	A set of numbers to define the range of channel 2 numerical values for which the color encode function will be defined. Only the extreme min and max values are necessary.
`colors.1`	The range of channel 1 colors to be returned by the function function. (default: c("#FFFFFF", "#FF0000"))
`colors.2`	The range of channel 2 colors to be returned by the function function. (default: c("#FFFFFF", "#0000FF"))
`combine_method`	(optional) For dual channel plots this is a string used to indicate how colors are combined to generate a two dimensional color scale. Options are "scaled_geomean" (same as "default"), "standard" (same as "euclidean"), "negative_euclidean", "mean", and "additive". See details.
`c1.fun`	(optional) A function to convert the numerical in channel 1 into colors. If not specified, this is generated based on `numbers.1` and `colors.1`.
`c2.fun`	(optional) A function to convert the numerical in channel 2 into colors. If not specified, this is generated based on `numbers.2` and `colors.2`.
`na.color`	(optional) The color returned from the function for NA values (default: "#CCC-CCC").

Value

makeTwoColorEncodeFunction() returns a function that takes 3 arguments and returns either a vector of hexadecimal colors or a 3-column matrix of columns. The arguments:

numbers.1	A vector of numbers for channel 1, to be encoded as a color value.
numbers.2	A vector of numbers for channel 2, to be encoded as a color value.
output_as	Specifies the type of return value. If 'vector' or 'rgb', the function returns a vector of hexadecimal colors (e.g."#FFCCAA"), if 'matrix','array', a three column numeric matrix is returned (Columns are "R", "G", or "B"). Currently, 'vector' are synonyms 'rgb', as are 'matrix' and 'array'

.

Examples

```
# Prepare the function:
twoColorEnc.fun <- makeTwoColorEncodeFunction( numbers.1 = c( 0.4, 6 ),
                                               numbers.2 = c(0.6, 20),
                                               colors.1 = c("white", "red"),
                                               colors.2 = c("white", "green" ),
                                               combine_method = "mean" )
# Encode two vectors of numbers as a single vector of colors:
colors_as_vector <- twoColorEnc.fun( numbers.1 = c( 0.4, 1.2, 5, 6 ),
                                     numbers.2 = c( 0.6, 6, 9, 20 ),
                                     output_as = 'vector' )
```

---

nzLog10 *nzLog10*

Description

Utility function to safely (non-zero) log10 transform p-values that are bounded at 0, and may be zero or may be rounded to zero in certain contexts. To get around this, prior to applying a log10 transformation the function adds a very small pseudocount to all the values if any are detected to be zero. This avoids the generation of negative infinities. (See details, below.)

Usage

```
nzLog10(x, quiet = FALSE)
```

Arguments

x	A numerical vector containing non-negative values.
quiet	A boolean that tells the script to suppress warning messages. (This does not suppress errors, however.)

Details

Prior to log10 transformation, this function first scans for any zeros in the input vector. If it finds any, it warns that zeros have been detected in the raw statistic, and that a pseudocount will be added. To do this the function assesses the precision of the numbers in the numerical vector by counting decimal places and determining the minimal non-zero number represented in the vector. It then takes whichever is the lesser of those numbers and adds a pseudocount equal to the lesser of 1/2 the precision, or 1/2 the lowest non-zero number.

Value

A vector containing transformed values.

Examples

```
p_vals <- c( 0.5, 0.001, 0.00001, 5e-19, 6.24e-23, 0 )
nzLog10( p_vals )
```

---

pick_MappedGeneSymbol *pick_MappedGeneSymbol*

Description

Function for matching values in `.from` vector derived from `Gene symbol` field from GEO feature data (e.g. "LOC101055758///LOC100041903///Gm2666///Gm7609///Csprs") with the first match in `.to` vector. The point of this is for a given differentially expressed feature, match the corresponding gene symbols to gene symbols present in a gene set collection. This (hopefully) leads to mapping more features in a GEO dataset to more gene symbols in a gene set collection to be searched. Symbol matches are done in a case independent way, and the value returned is the value in the .to vector (with its particular capitalization), such that pathways analysis can be easily performed.

Usage

```
pick_MappedGeneSymbol(.from, .to)
```

Arguments

`.from`	Character vector containing concatenated, triple-slash delimited gene symbols (e.g. "LOC101055758///LOC100041903///Gm2666///Gm7609///Csprs")
`.to`	Character vector conrtaining gene symbols to be matched (e.g. "Gm2666")

Value

A vector containing the matched symbols.

Examples

```
 ## Not run:
library(GSNA)
library(GEOquery)
library(tmod)

gset <- getGEO("GSE75203", GSEMatrix =TRUE, AnnotGPL=TRUE)
GSE75203.fdata <- fData(gset$GSE75203_series_matrix.txt.gz)
msig <- tmodImportMSigDB( file = file.path( "msigdb_v7.5.1.xml.gz" ) )

GSE75203.fdata$MappedGeneSymbol <- pick_MappedGeneSymbol( .from = GSE75203.fdata$`Gene symbol`,
                                                          .to = msig$GENES$ID )

## End(Not run)
```

`plot.GSNData` *plot plot.GSNData*

Description

Plot method for the networks within GSNData objects, implemented with `gsnPlotNetwork`.

Usage

```
## S3 method for class 'GSNData'
plot(x, ...)
```

Arguments

`x`	A GSNData object containing a pared distance matrix with edges.
`...`	Arguments passed on to `gsnPlotNetwork`

`object` A GSNData object containing a pared distance matrix with edges. NOTE: when calling as `plot.GSNData`, use the argument `x` instead.

`pathways_dat` (optional) data.frame containing associated pathways data. This defaults to whatever pathways data has already been imported into this GSNData object in `object$pathways$data`.

`distance` (optional) The name of a distance metric used, defaults to whatever `default_distance` is.

`id_col` (optional) This is the name of the column in the pathways data.frame that corresponds to the names of gene sets. The default value is specified by `object$pathways$id_col`. (See details.)

`substitute_id_col` (optional) This is the name of the column that is to be substituted for the `id_col` column when labeling network vertices. (See details.)

`stat_col` (optional) This is the name of the column in the pathways data.frame that contains a significance value for coloring network vertices. The default value is specified by `object$pathways$stat_col`.

stat_col_2 (optional) This is the name of an optional second column in the pathways data.frame that contains a significance value for coloring network vertices in a 2-color network. The default value is specified by object$pathways$stat_col_2. When specified, a 2-color network is generated. To force a 2-color network to plot as a standard 1-color network using stat_col alone, use stat_col_2 = NA.

sig_order (optional) This indicates the behavior of stat_col, whether low values ('loToHi') or high values ('hiToLo') are most significant. The default value is specified in object$pathways$sig_order.

sig_order_2 (optional) This indicates the behavior of stat_col_2, whether low values ('loToHi') or high values ('hiToLo') are most significant. The default value is specified in object$pathways$sig_order_2.

n_col (optional) This is the name of the column in the pathways data.frame that contains a value for gene set size, or any other value intended to be the bases of leaf scaling. When specified, leaf sizes will be scaled by this value, either as a function argument, or in the object$pathways$n_col field. An NA value can be used to override the the value in object$pathways$n_col and suppress leaf scaling when n_col has been set in the object. (default is the value in object$pathways$n_col).

optimal_extreme (optional) This indicates the behavior of the statistic used to generate the distance metric, specifically whether low values ('min') or high values 'max' are to be regarded as close. This is used for scaling the width and the color of the edges connecting vertices. See scale.edges.by.distance, below: (default: object$distances[distance]$pared_optimal_extreme or if that's NULL, object$distances[distance]$optimal_extreme)

transform_function (optional) This is a function to transform the values in stat_col so that they are suitable for amenable to color-scaling. For *p*-values, a log transformation is often useful, but can produce negative infinities if the transformation is applied to zero. By default the function is the nzLog10 (non-zero log10) function, provided by this package, which adds a small pseudocount to p-values when log10 transforming values equal to zero. If values in stat_col are less than zero, then log10 transformation is inappropriate and will introduce NAs, and therefore some other method should be used. (default: nzLog10)

pathways_title_col (optional) Indicates a column to be used as the 'Title' column for network vertices. If unset, the function attempts to search for a title column from the following values: c("Title", "Name", "NAME", "STANDARD_NAME") (See details.)

edge_colors (optional) A vector of colors included to generate a scale represent the numerical value of the edge distances. By default, the colors are arranged as a rainbow with black and purple representing the greatest distannces, and orange and red the nearest distances. This feature (and argument) will likely be deprecated in future versions. (default: edge_colors = c("black", "purple", "blue", "green","yellow4", "orange","red"))

vertex_colors (optional) This is the standard set of colors used for a standard single color network. By default, c("white","yellow","red") is used, coloring low values white, high values red, and intermediate values yellow if sig_order is "loToHi" and vice versa if sig_order is "hiToLo".

vertex_colors.1 (optional) This is the range of colors used for a 2-color network corresponding to values of stat_col. Up to 2 colors can be used, and should correspond to a color contrasting with vertex_colors.2. The

default is c("white","red"), coloring high values red and low values white if sig_order is "loToHi" and vice versa if sig_order is "hiToLo".

vertex_colors.2 (optional) This is the range of colors used for a 2-color network corresponding to values of stat_col_2. Up to 2 colors can be used, and should correspond to a color contrasting with vertex_colors.2. The default is c("white","blue"), coloring high values blue and low values white if sig_order_2 is "loToHi" and vice versa if sig_order is "hiToLo".

combine_method (optional) For dual channel plots this is a string used to indicate how colors are combined to generate a two dimensional color scale. Options are "scaled_geomean" (same as "default"), "standard" (same as "euclidean"), "negative_euclidean", "mean", and "additive". See details.

na.color (optional) This color is assigned to vertices for which there is an NA value. (default: "#CCCCCC")

filename (optional) An output file name for the plot. If 'out_format' is not set (see below), the output file type will be determined by the file suffix, which can be '.svg', '.pdf', or '.png'. If the out_format cannot be determined from the file name, than it may be manually set with out_format. If the output file type cannot be determined from the filename or out_format arguments, an error will be thrown.

out_format (optional) Output filetype when filename is specified, either 'svg', 'png', 'pdf', or 'plot' (default if filename is not specified). For more information, see Details.

width (optional) Sets the width of the output canvas in inches. Defaults to the width of the present graphical device.

height (optiona) Sets the height of the output canvas in inches. Defailts to the height of the present graphical device.

vertex.shape (optional) Shape of the vertex, passed to igraph::plot.igraph. By default, the value is 'circle'.

vertex.size (optional) Size of vertices, passed to igraph::plot.igraph. By default, the value is NULL, and the function attempts to pick a reasonable value based on the canvas size and the number of gene sets.

vertex.size.range (optional) The range of vertex sizes used in plots, from low to high. This is used when n_col is specified and vertex sizes are inended to be scaled. If this is not specified, then the function attempts to select appropriate values based on size of the figure being generated.

vertex.label.cex (optional) Size of vertex labels, passed to igraph::plot.igraph. As with vertex.size, by default, the value is NULL, and the function attempts to pick a reasonable value based on the canvas size and the number of gene sets.

vertex.label.col (optional) Color of vertex labels, passed to igraph::plot.igraph. If not specified, the function attempts to pick a contrasting color for vertex label text using the contrasting_color.fun argument. (default: NULL)

vertex.frame.color (optional) Color of the vertex border. (default par('fg'))

contrasting_color.fun (optional) A function to pick a color for vertex labels that contrasts with the vertex fill color. If unspecified, the function attempts to pick a suitable function for generating suitable set of contrasting colors, based on the contrasting_color() function. (default: For single channel plots using color scales defined with vertex_colors, or dual channel color scales defined with vertex_colors.1, or vertex_colors.2 using yellow

or orange, `contrasting_color(type="binary")` is used, and otherwise `contrasting_color(type="blackyellow")` is used.)

`scale_labels_by_vertex` (optional) Logical that tells the function to scale the text in vertex labels by the size of the vertex. (default: TRUE)

`max_edge_width` (optional) Size of vertex labels, passed to `igraph::plot.igraph`. By default, the value is NULL, and the function attempts to pick a reasonable value based on the canvas size and the number of gene sets.

`scale.edges.by.distance` (optional) A logical telling the function to scale edges between vertices on the basis of distance. NOTE: If `optimal_extreme == "max"`, then smaller numbers are treated as more distant, and conversely if `optimal_extreme == "min"`, larger numbers are treated as more distant. (default: FALSE)

`color.edges.by.distance` (optional) A logical telling the function to color edges between vertices on the basis of distance. This functionality will likely be deprecated. (default: FALSE)

`edge_arrow_size` (optional) Size of vertex labels, passed to `igraph::plot.igraph`. By default, the value is NULL, and the function attempts to pick a reasonable value based on the canvas size and the number of gene sets.

`seed` (optional) This is a seed that the function uses to generate a plot layout. By default it is 29189892, and this results in a repeatable behavior for plots. However, to randomize the plot layout behavior, this value may be set to NULL, or if some other repeatable layout is desired, another seed may be used.

`layout` (optional) Either a function that generates a layout or a numerical matrix containing a vertex layout with two columns corresponding to *x* and *y* coordinates. This argument is passed to the `igraph` plot method that is subsequently called by `gsnPlotNetwork.old()` (see `.plot`, below). The default `layout` is the anonymous function `function(x){igraph::layout_with_fr(x, grid = "nogrid" )}`, which calls `igraph::layout_with_fr()` (implementing Fruchterman-Reingold layout) with the `grid="nogrid"` option, enabling proper layout of networks with >= 1000 gene set vertices. Other useful layouts for `igraph` networks include `igraph::layout_with_fr` (default Fruchterman-Reingold), `igraph::layout_with_dh` (implementing Davidson-Harel layout), `igraph::layout_as_tree`, `igraph::layout_nicely`, and others. For more details about layouts, see `igraph.plotting`.

`.plot` (optional) A plot function used to render the internally generated `igraph` object. By default `igraph::plot.igraph` is used, but for interactive plotting, `igraph::tkplot` may be used. For more details about plotting, see `igraph.plotting`.

`show.legend` (optional) A logical value telling the function whether or not to show legends. Legends for vertex size and node color are currently supported. (default: TRUE)

`legend.lab.cex` (optional) The font size of legend label text as cex units. If not specified, the function will attempt to pick an appropriate value based on the figure layout.

`legend.axis.cex` (optional) The font size of legend axis text as cex units. If not specified, the function will attempt to pick an appropriate value based on the figure layout.

`legend.fg` (optional) The foreground color of the legend that controls the color of text, axes, axis labels, ticks, and legend border. (default: par('fg'))

`legend.bg` (optional) The background color of the legend. This argument doesn't currently work, and may be removed in the future. (default: "#CCCCCC")

`legend.vertex.fg` (optional) The border color of vertices for vertex size legends. This argument allows the legend vertex frame color to be set separately from vertex.frame.color. (default: vertex.frame.color)

`legend.vertex.bg` (optional) The fill color of vertices for vertex size legends. (default: "#DDDDDD")

`font_face` (optional) The font face used for the figure. (default: par("family"))

`main` (optional) The plot title. (default: NULL)

`cex.main` (optional) The font size in cex units of the main title. (default: 1.5 * par('cex'))

`mar.main` (optional) The number of lines set aside for a main title when main is used. (default: 3.2)

`lines.main` (optional) The distance of the main title in lines from the top of the plot. (default: 0.9)

`.mar.plot` (optional) The margins of the plot itself. If unnspecified, the function will attempt to reserve enough room to the right of the plot for the legend or legends.

`draw.legend.box.bool` (option) Logical indicating whether bounding boxes should be drawn for the legends.

`legend.free.cex.bool` (optional) Logical allowing independent optimized sizing of legend label font sizes if TRUE. (default: FALSE)

`legend_x_size.in` (optional) The width of the legend in inches. If not set, the function attempts to choose an appropriate value. (default: `min(2,max(width*2/5,width-heig`

`colors.n` (optional) The number of colors in for each channel in 1 or 2 channel plots. For single channel plots the number of colors is simply equal to this number. For dual channel plots the total number of colors in the legend is equal to the square of this number. (default: 100)

`new` (optional) Logical telling the function (if true) that a new plot should be added to an existing device (if TRUE) or that the current device should be cleared and written over (if FALSE). (default: FALSE)

`legend_spacing.x.in` (optional) Space between plot and legend in inches. This can be used to adjust the horizontal position and move the legend closer to or farther away from the plot region. Since the netork plot may not fill the entire plotting region, it may be useful to use negative values to move the legends closer to the plot. (default: 2 character widths)

`legend_spacing.y.in` (optional) Space between legends in inches. (default: 1 character height)

`DO_BROWSER` (option) Logical indicating whether browser() should be run for this function. (For debugging purposes, will probably remove.)

Details

This function is primarily for taking `GSNData` object containing a distance matrix, an associated `edges` edge-list and pathways data, and generating and rendering a corresponding `igraph` object. The function attempts to plot the corresponding network with vertices labeled with a gene set `ID` and corresponding `Title`, and colored according to the significance values represented in `stat_col` using `sig_order` as an indicator of whether high or low values are more significant. Edges are scaled by the value of the value of the distance statistic in the pared distance matrix.

When the parameters vertex.shape, vertex.size, vertex.label.cex, max_edge_width, and edge_arrow_size are not specified, the function attempts to pick reasonable values. These parameters are assembled into a list and attached to the returned igraph object as an attribute named GSNA_plot_params. To optimize plots, the user can examine these parameters by calling the following on the output of the function:

```
attr( x = nw.igraph, which = "GSNA_plot_params" )
```

Value

An igraph network object is returned, invisibly.

See Also

gsnPlotNetwork gsnToIgraph plot.igraph

Examples

```
## Not run:
gsnPlotNetwork.old( object = analysis.GSN )

## End(Not run)
```

---

plot_GSNData *plot_GSNData*

Description

S3 method to plot GSNData objects. It is a wrapper for gsnPlotNetwork and works very similarly.

Usage

```
plot_GSNData(
  x,
  y = NULL,
  object,
  pathways_dat = NULL,
  distance = NULL,
  id_col = NULL,
  substitute_id_col = NULL,
  stat_col = NULL,
  stat_col_2 = NULL,
  sig_order = NULL,
  sig_order_2 = NULL,
  optimal_extreme = NULL,
  transform_function = nzLog10,
  pathways_title_col = "Title",
 edge_colors = c("black", "purple", "blue", "green", "yellow4", "orange", "red"),
  vertex_colors = c("white", "yellow", "red"),
  vertex_colors.1 = c("white", "red"),
  vertex_colors.2 = c("white", "blue"),
  filename = NULL,
```

```
  out_format = NULL,
  width = NULL,
  height = NULL,
  vertex.shape = "circle",
  vertex.size = NULL,
  vertex.label.cex = NULL,
  max_edge_width = NULL,
  edge_arrow_size = NULL,
  seed = 29189892,
  layout = function(x) {
     igraph::layout_with_fr(x, grid = "nogrid")
 },
  .plot = igraph::plot.igraph,
  plot_layout_params_bool = FALSE,
  ...
)
```

Arguments

x	A GSNData class object corresponding to the object parameter of gsnPlotNetwork.
y	Currently ignored, but needed for compatibility with the plot generic method.
object	A GSNData object containing a pared distance matrix with edges. NOTE: when calling as plot.GSNData, use the argument x instead.
pathways_dat	(optional) data.frame containing associated pathways data. This defaults to whatever pathways data has already been imported into this GSNData object in object$pathways$data.
distance	(optional) The name of a distance metric used, defaults to whatever default_distance is.
id_col	(optional) This is the name of the column in the pathways data.frame that corresponds to the names of gene sets. The default value is specified by object$pathways$id_col. (See details.)
substitute_id_col	(optional) This is the name of the column that is to be substituted for the id_col column when labeling network vertices. (See details.)
stat_col	(optional) This is the name of the column in the pathways data.frame that contains a significance value for coloring network vertices. The default value is specified by object$pathways$stat_col.
stat_col_2	(optional) This is the name of an optional second column in the pathways data.frame that contains a significance value for coloring network vertices in a 2-color network. The default value is specified by object$pathways$stat_col_2. When specified, a 2-color network is generated. To force a 2-color network to plot as a standard 1-color network using stat_col alone, use stat_col_2 = NA.
sig_order	(optional) This indicates the behavior of stat_col, whether low values ('loToHi') or high values ('hiToLo') are most significant. The default value is specified in object$pathways$sig_order.
sig_order_2	(optional) This indicates the behavior of stat_col_2, whether low values ('loToHi') or high values ('hiToLo') are most significant. The default value is specified in object$pathways$sig_order_2.

optimal_extreme
: (optional) This indicates the behavior of the statistic used to generate the distance metric, specifically whether low values ('min') or high values 'max' are to be regarded as close. This is used for scaling the width and the color of the edges connecting vertices. See scale.edges.by.distance, below: (default: object$distances[distance]$pared_optimal_extreme or if that's NULL, object$distances[distance]$optimal_extreme)

transform_function
: (optional) This is a function to transform the values in stat_col so that they are suitable for amenable to color-scaling. For *p*-values, a log transformation is often useful, but can produce negative infinities if the transformation is applied to zero. By default the function is the nzLog10 (non-zero log10) function, provided by this package, which adds a small pseudocount to p-values when log10 transforming values equal to zero. If values in stat_col are less than zero, then log10 transformation is inappropriate and will introduce NAs, and therefore some other method should be used. (default: nzLog10)

pathways_title_col
: (optional) Indicates a column to be used as the 'Title' column for network vertices. If unset, the function attempts to search for a title column from the following values: c("Title", "Name", "NAME", "STANDARD_NAME") (See details.)

edge_colors
: (optional) A vector of colors included to generate a scale represent the numerical value of the edge distances. By default, the colors are arranged as a rainbow with black and purple representing the greatest distannces, and orange and red the nearest distances. This feature (and argument) will likely be deprecated in future versions. (default: edge_colors = c("black", "purple", "blue", "green","yellow4", "orange","red"))

vertex_colors
: (optional) This is the standard set of colors used for a standard single color network. By default, c("white","yellow","red") is used, coloring low values white, high values red, and intermediate values yellow if sig_order is "loToHi" and vice versa if sig_order is "hiToLo".

vertex_colors.1
: (optional) This is the range of colors used for a 2-color network corresponding to values of stat_col. Up to 2 colors can be used, and should correspond to a color contrasting with vertex_colors.2. The default is c("white","red"), coloring high values red and low values white if sig_order is "loToHi" and vice versa if sig_order is "hiToLo".

vertex_colors.2
: (optional) This is the range of colors used for a 2-color network corresponding to values of stat_col_2. Up to 2 colors can be used, and should correspond to a color contrasting with vertex_colors.2. The default is c("white","blue"), coloring high values blue and low values white if sig_order_2 is "loToHi" and vice versa if sig_order is "hiToLo".

filename
: (optional) An output file name for the plot. If 'out_format' is not set (see below), the output file type will be determined by the file suffix, which can be '.svg', '.pdf', or '.png'. If the out_format cannot be determined from the file name, than it may be manually set with out_format. If the output file type cannot be determined from the filename or out_format arguments, an error will be thrown.

out_format
: (optional) Output filetype when filename is specified, either 'svg', 'png', 'pdf', or 'plot' (default if filename is not specified). For more information, see Details.

`width` (optional) Sets the width of the output canvas in inches. Defaults to the width of the present graphical device.

`height` (optiona) Sets the height of the output canvas in inches. Defailts to the height of the present graphical device.

`vertex.shape` (optional) Shape of the vertex, passed to `igraph::plot.igraph`. By default, the value is `'circle'`.

`vertex.size` (optional) Size of vertices, passed to `igraph::plot.igraph`. By default, the value is NULL, and the function attempts to pick a reasonable value based on the canvas size and the number of gene sets.

`vertex.label.cex` (optional) Size of vertex labels, passed to `igraph::plot.igraph`. As with vertex.size, by default, the value is NULL, and the function attempts to pick a reasonable value based on the canvas size and the number of gene sets.

`max_edge_width` (optional) Size of vertex labels, passed to `igraph::plot.igraph`. By default, the value is NULL, and the function attempts to pick a reasonable value based on the canvas size and the number of gene sets.

`edge_arrow_size` (optional) Size of vertex labels, passed to `igraph::plot.igraph`. By default, the value is NULL, and the function attempts to pick a reasonable value based on the canvas size and the number of gene sets.

`seed` (optional) This is a seed that the function uses to generate a plot layout. By default it is 29189892, and this results in a repeatable behavior for plots. However, to randomize the plot layout behavior, this value may be set to NULL, or if some other repeatable layout is desired, another seed may be used.

`layout` (optional) Either a function that generates a layout or a numerical matrix containing a vertex layout with two columns corresponding to *x* and *y* coordinates. This argument is passed to the `igraph` plot method that is subsequently called by `gsnPlotNetwork.old()` (see `.plot`, below). The default `layout` is the anonymous function `function(x){igraph::layout_with_fr(x, grid = "nogrid" )}`, which calls `igraph::layout_with_fr()` (implementing Fruchterman-Reingold layout) with the `grid="nogrid"` option, enabling proper layout of networks with >= 1000 gene set vertices. Other useful layouts for `igraph` networks include `igraph::layout_with_fr` (default Fruchterman-Reingold), `igraph::layout_with_dh` (implementing Davidson-Harel layout), `igraph::layout_as_tree`, `igraph::layout_nicely`, and others. For more details about layouts, see `igraph.plotting`.

`.plot` (optional) A plot function used to render the internally generated `igraph` object. By default `igraph::plot.igraph` is used, but for interactive plotting, `igraph::tkplot` may be used. For more details about plotting, see `igraph.plotting`.

`plot_layout_params_bool` (optional) A boolean value that changes the return value of the function to a list containing plotting parameters when set to `TRUE` (see below). The default is `FALSE`.

`...` Additional parameters, currently ignored.

Details

This method is a wrapper for `gsnPlotNetwork()`. It is primarily for taking `GSNData` object containing a distance matrix, an associated `edges` edge-list and pathways data, and generating and rendering a corresponding `igraph` object. The function attempts to plot the corresponding network with vertices labeled with a gene set `ID` and corresponding `Title`, and colored according to the

significance values represented in stat_col using sig_order as an indicator of whether high or low values are more significant. Edges are scaled by the value of the value of the distance statistic in the pared distance matrix.

When the parameters vertex.shape, vertex.size, vertex.label.cex, max_edge_width, and edge_arrow_size are not specified, the function attempts to pick reasonable values. These parameters are assembled into a list and attached to the returned igraph object as an attribute named GSNA_plot_params. To optimize plots, the user can examine these parameters by calling the following on the output of the function:

```
attr( x = nw.igraph, which = "GSNA_plot_params" )
```

Value

An igraph network object is returned, invisibly, unless the plot_layout_params_bool is set as TRUE in which case, the list of plotting parameters contained in the igraph object's GSNA_plot_params attribute is returned instead.

See Also

gsnPlotNetwork gsnToIgraph plot.igraph

Examples

```
## Not run:
plot( x = analysis.GSN )

## End(Not run)
```

---

print.GSNData *print.GSNData*

Description

Print a short description of a GSNData object.

Usage

```
## S3 method for class 'GSNData'
print(object)
```

Arguments

object	A GSNData object.

Value

Invisibly returns the GSNData object.

pw_id_col *pw_id_col*

Description

Retrieve or set the pathways id_col field in a GSNData object.

Usage

```
pw_id_col(object)

pw_id_col(object) <- value
```

Arguments

object	An GSNData object.
value	A character vector of length 1 containing the name of a column within the pathways data to be used as a gene set identifier. The GSNData object must contain a pathways data data.frame, or else an error will be thrown.

Value

The name of the id_col field.

A GSNData object with the value of the $pathways$id_col field set.

See Also

gsn_distances

Examples

```
## Not run:
# Print the value of the default_distance:
pw_id_col( analysis.GSN )

## End(Not run)
## Not run:
# Set the value of the id_col to 'ID':
pw_id_col( analysis.GSN ) <- 'ID'

## End(Not run)
```

pw_n_col *pw_n_col*

Description

Retrieve or set the pathways n_col field in a GSNData object.

Usage

```
pw_n_col(object)

pw_n_col(object) <- value
```

Arguments

object	An GSNData object.
value	A character vector of length 1 containing the name of a column within the pathways data to be used as a gene set identifier. The GSNData object must contain a pathways data data.frame, or else an error will be thrown.

Value

The name of the n_col field.

A GSNData object with the value of the $pathways$n_col field set.

See Also

gsn_distances

Examples

```
## Not run:
# Print the value of the default_distance:
pw_n_col( analysis.GSN )

## End(Not run)
## Not run:
# Set the value of the n_col to 'SIZE':
pw_n_col( analysis.GSN ) <- 'SIZE'

## End(Not run)
```

pw_sig_order *pw_sig_order*

Description

Retrieve or set the pathways sig_order field in a GSNData object.

Usage

```
pw_sig_order(object)

pw_sig_order(object) <- value
```

Arguments

object	An GSNData object.
value	A character vector of length 1 containing the name of a column within the pathways data to be used as a gene set identifier. The GSNData object must contain a pathways data data.frame, or else an error will be thrown. Valid values are 'hiToLo', and 'loToHi'.

Value

The name of the sig_order field.

A GSNData object with the value of the $pathways$sig_order field set.

See Also

gsn_distances

Examples

```
## Not run:
# Print the value of the default_distance:
pw_sig_order( analysis.GSN )

## End(Not run)
## Not run:
# Set the value of the sig_order to 'ID':
pw_sig_order( analysis.GSN ) <- 'ID'

## End(Not run)
```

pw_sig_order_2 *pw_sig_order_2*

Description

Retrieve or set the pathways sig_order_2 field in a GSNData object.

Usage

```
pw_sig_order_2(object)

pw_sig_order_2(object) <- value
```

Arguments

object	An GSNData object.
value	A character vector of length 1 containing the name of a column within the pathways data to be used as a gene set identifier. The GSNData object must contain a pathways data data.frame, or else an error will be thrown. Valid values are 'hiToLo', and 'loToHi'.

Value

The name of the sig_order_2 field.

A GSNData object with the value of the $pathways$sig_order_2 field set.

See Also

gsn_distances

Examples

```
## Not run:
# Print the value of the default_distance:
pw_sig_order_2( analysis.GSN )

## End(Not run)
## Not run:
# Set the value of the sig_order_2 to 'ID':
pw_sig_order_2( analysis.GSN ) <- 'ID'

## End(Not run)
```

pw_stat_col *pw_stat_col*

Description

Retrieve or set the pathways stat_col field in a GSNData object.

Usage

```
pw_stat_col(object)

pw_stat_col(object) <- value
```

Arguments

object	An GSNData object.
value	A character vector of length 1 containing the name of a column within the pathways data to be used as a gene set identifier. The GSNData object must contain a pathways data data.frame, or else an error will be thrown.

Value

The name of the stat_col field.

A GSNData object with the value of the $pathways$stat_col field set.

See Also

gsn_distances

Examples

```
## Not run:
# Print the value of the default_distance:
pw_stat_col( analysis.GSN )

## End(Not run)
## Not run:
# Set the value of the stat_col to 'ID':
pw_stat_col( analysis.GSN ) <- 'ID'

## End(Not run)
```

pw_stat_col_2 *pw_stat_col_2*

Description

Retrieve or set the pathways stat_col_2 field in a GSNData object.

Usage

```
pw_stat_col_2(object)

pw_stat_col_2(object) <- value
```

Arguments

object	An GSNData object.
value	A character vector of length 1 containing the name of a column within the pathways data to be used as a gene set identifier. The GSNData object must contain a pathways data data.frame, or else an error will be thrown.

Value

The name of the stat_col_2 field.

A GSNData object with the value of the $pathways$stat_col_2 field set.

See Also

gsn_distances

Examples

```
## Not run:
# Print the value of the default_distance:
pw_stat_col_2( analysis.GSN )

## End(Not run)
## Not run:
# Set the value of the stat_col_2 to 'ID':
pw_stat_col_2( analysis.GSN ) <- 'ID'

## End(Not run)
```

pw_type *pw_type*

Description

Retrieve or set the pathways type field in a GSNData object.

Usage

```
pw_type(object)

pw_type(object) <- value
```

Arguments

object	An GSNData object.
value	A character vector of length 1 containing the name of a column within the pathways data to be used as a gene set identifier. The GSNData object must contain a pathways data data.frame, or else an error will be thrown.

Value

The name of the type field.

A GSNData object with the value of the $pathways$type field set.

See Also

gsn_distances

Examples

```
## Not run:
# Print the value of the default_distance:
pw_type( analysis.GSN )

## End(Not run)
## Not run:
# Set the value of the type to 'ID':
pw_type( analysis.GSN ) <- 'ID'

## End(Not run)
```

read_david_data_file *read_david_data_file*

Description

Parses a text file output by the DAVID web application (https://david.ncifcrf.gov/) (see details).

Usage

```
read_david_data_file(file, output = "flat", redundant = FALSE, sep = "\t")
```

Arguments

file	A file path pointing to a DAVID "Functional Annotation Cluster" or "Functional Annotation Chart" text file.
output	(optional) Specifies the type of output. (default "flat") This parameter can take one of three values: **"flat":** If "flat" is specified, a single data.frame containing the standard DAVID output fields is returned. For "Functional Annnotation Cluster" data, an additional column named `` `Cluster (ES)` `` is included, containing for each gene set, comma-separated DAVID `` `Annotation Cluster` `` assignments and in parentheses, DAVID Enrichment Scores. **"hierarchic":** For "hierarchic" output, a list containing a set of data.frames for each `` `Annotation Cluster` `` is returned. This only works with "Functional Annotation Cluster" output. **"GSC":** DAVID data sets contain nested gene sets in their `` `Genes` `` column. The gene sets can be extracted as a list of gene set vectors by specifying this option.
redundant	(optional) The "Functional Annotation Cluster" output of DAVID contains fuzzy DAVID clusters in which a given gene set may be assigned to multiple clusters. As a result, some gene sets can have multiple lines in a "Functional Annotation Cluster" output file, resulting in redundant data.frame rows. If this value is FALSE, the returned "flat" data.frame will have gene set duplicates removed and the DAVID `` `Annotation Cluster` `` identities of each gene set listed as comma separated values in the `` `Cluster (ES)` `` column. If TRUE than the redundancies are tolerated and replicate gene set rows are not collapsed. (default: FALSE)
sep	(optional) Specifies the separator used in the DAVID output file. This probably does not need to be specified. (default "\t")

Details

This function parses tab-separated text files from the DAVID web application (https://david.ncifcrf.gov/). Two variants of DAVID output are supported, specifically the data format generated by selecting "Functional Annotation Chart" or "Functional Annotation Cluster" and downloading the resulting data as a text file.

The parser expects the following fields in the data: "Category", "Term", "Count", "%", "PValue", "Genes", "List Total", "Pop Hits", "Pop Total", "Fold Enrichment", "Bonferroni", "Benjamini", and "FDR".

To create a data.framme suitable for use with `gsnAddPathwaysData()`, the default options are required, partucularly `output = "flat"` and `redundant = FALSE`.

Value

The function returns either a data.frame containing DAVID data, a list of data.frames, or a list of gene sets. (see documentation for the `output` parameter above).

See Also

`gsnImportDAVID()`

Examples

```
## Not run:
    file_path = '/path/to/david_FAC.txt'
    # Get a data.frame containing DAVID results.
    pathways.david <- read_david_data_file( file = file_path )
    # Extract a gene set collection from the DAVID results file:
    pathways.GSC <- read_david_data_file( file = file_path,
                                          output = "GSC" )

## End(Not run)
```

`read_gmt` *read_gmt*

Description

This function parses a GMT file, documented here.

Usage

```
read_gmt(file)
```

Arguments

`file`	The path to GMT file to parse.

Value

This returns a GSC (gene set collection) as a name list of vectors, where the names correspond to gene set identifiers and the vectors are gene symbols.

See Also

`gsc2tmod()`

Examples

```
## Not run:
gsc <- read_gmt( "gene_set_collection.GMT" )

## End(Not run)
```

`recursiveGetConnectedVertices` *recursiveGetConnectedVertices*

Description

Internal function used by the `assignSubnets()` function. When passed the name of a vertex, this function recursively retrieves a full set of connected vertices to populate a character vector named `vertices.v`. The function requires that the `edges.df` data.frame containing an edge list and the `vertices.v` vector, which it populates with the names of connected verticies, to be present in the environment passed using the `e` argument. The default environment is the calling environment.

Usage

```
recursiveGetConnectedVertices(vertx, e = environment())
```

Arguments

`vertx`	Name of a vertex.
`e`	An environment containing an edge list data.frame (`edges.df`) and a vertices.v character vector.

Details

This method is currently of limited use outside of assignSubnets, for which it does most of the work.

`scoreJaccardMatrix_C` *scoreJaccardMatrix_C*

Description

Takes a presence/absence matrix with genes as the rows and modules as columns and calculates a matrix of Jaccard index values.

Usage

```
scoreJaccardMatrix_C(geneSetCollection_m)
```

Arguments

`geneSetCollection_m`
: (required) A logical presence/absence matrix representation of a gene set collection in which columns correspond to gene sets, rows correspond to genes and values are `TRUE` if a gene is present in a gene set and `FALSE` otherwise. Row and column names correspond to gene symbols and gene set identifiers, respectively. NOTE: for a typical GSNA analysis, this matrix would include only observerved filtered genes and significant gene set hits from pathways analysis. Using a matrix version of the full MSigDB without filtering geneas, for example, would likely be unworkably slow and memory intensive.

Details

The Jaccard index J for two sets A and B is defined as:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Value

This function returns a matrix of Jaccard index values between gene modules. Values on the diagonal corresponding to self-Jaccard indices are returned as NA.

See Also

`buildGeneSetNetworkJaccard()` `scoreLFMatrix_C()`

Examples

```
## Not run:
library( GSNA )
jaccardMatrix <- scoreJaccardMatrix_C( PresenceAbsMatrix )

## End(Not run)
```

---

scoreLFMatrix_C *scoreLFMatrix_C*

Description

Takes a presence/absence matrix with genes as the rows and modules as columns and calculates a matrix of log-transformed Fisher *p*-values.

Usage

```
scoreLFMatrix_C( geneSetCollection_m,
                 e_precision = 12,
                 alternative = 1 )
```

Arguments

`geneSetCollection_m`
: (required) A logical presence/absence matrix representation of a gene set collection in which columns correspond to gene sets, rows correspond to genes and values are `TRUE` if a gene is present in a gene set and `FALSE` otherwise. Row and column names correspond to gene symbols and gene set identifiers, respectively. NOTE: for a typical GSNA analysis, this matrix would include only observerved filtered genes and significant gene set hits from pathways analysis. Using a matrix version of the full MSigDB without filtering geneas, for example, would likely be unworkably slow and memory intensive.

`e_precision`
: (optional, default 12) Numeric to control the precision of the log p-value calculated. Due to precision limits inherant in C++ double precision numbers, log p-values for which the corresponding untransformed p-values differ by more than a certain magnitude cannot effectively be added. This feature was introduced as a way to accelerate summation of p-values so as to allow summation to be cut off when the acceptable level of precision had been reached, but it was found that it also seems to prevent artifacts caused by numerical underruns.

`alternative`
: (optional, default 1) An interger value specifying one of 4 alternative p-value calculations where `1` specifies single, upper tail log Fisher p-value, `2` signifies single, lower-tail Fisher p-value, `3` signifies 2-tailed Fisher p-value, and `4` signifies partial Fisher p-value (see below).

Details

Fisher *p*-values have long been used to assess the statistical significance of over- or underrepresentation of a component of a mixture to assess whether a sample is drawn from a particular mixture. The test has also long been used in pathways analysis as a way to assess whether an experimentally derived list of genes contains a statistical overrepresentation of genes from predefined gene sets or modules. Such experimental gene lists may include differentially expressed genes from a transcriptomic experiment, genes posessing promoters with differential chromatin accessibility from an ATACSeq experiments, genes that were positive in screens of mutants, genes that were identified from GWAS experiments, and genes from other analyses. Likewise, the gene sets or modules are generally drawn from databases of experimentally characterized pathways, sets of genes over- or under-expressed in particular conditions, or associated with particular biological processes, chromosome regions, etc.

In the case of GSNA, we use the Fisher test to assess the overlap of genes not between an experimentally derived gene list and predefined gene sets from a database, but between the predefined gene sets themselves given their observability in a particular experiment.

Value

A numerical matrix containing the specified log Fisher *p*-values for all non-self pairs. Values on the diagonal (which would correspond to self-self comparison *p*-values) are NA. The `'lower_is_closer'` attribute on the matrix is set to `TRUE`, except in the case of `alternative=2` where it is set to `FALSE`.

The `distance` attribute in the output matrix is set to `'stlf'` for option 1 (single, upper tail), `'ltlf'` for option 2 (lower tail), `'ttlf'` for option 3 (two-tailed), and `'lf'` for option 4 (log partial Fisher *p*-value).

Implementation

We use the Fisher test to assess the statistical significance of the overlap of two gene sets. For our purposes the test determines whether two gene sets share a greater (or in some cases less) than

expected number of common members, assuming a null hypothesis of random membership. The two sets need not necessarily be of the same size, but are for the purposes of the test assumed to have set sizes.

Consider a 2x2 contingency matrix of the following form:

```
| a  b |
| c  d |
```

Given a background of observable genes and two gene sets, *i* and *j* that may overlap, this contingincy table is used to represent four numbers:

- *a*: the number of genes observed in the background but not in *i* or *j*
- *b*: the number of observed genes in *i* but not *j*
- *c*: the number of observed genes in *j* but not *i* and
- *d*: the number of observed genes in both *j* and *i*, i.e. the overlap.

The *partial*-Fisher *p*-value, signifying the likelihood of that particular contingency table is given by:

$$p = \frac{(a+b)!\,(c+d)!\,(a+c)!\,(b+d)!}{a!\,b!\,c!\,d!\,(a+b+c+d)!}$$

This partial *p*-value is what is returned in the distance matrix when the artgument `alternative = 4` and it is less than, though tracks closely with, the two-tailed p-value, in most cases.

The actual single- and two-tailed *p*-values are derived from this number by summation, keeping the sum of each row and column of the 2x2 contingency matrix constant, as per the assumptions of the Fisher test. For the single-tailed alternative representing the upper-tail 'greater-than' expected overlap of the two gene sets (`alternative = 1`), the terms start with *d* as the observed number of shared members between set *i* and set *j*. Then *d* is incremented toward the maximal number possible shared genes (the lesser of the number of genes in sets *i* and *j*). *a*, *b*, and *c* adjusted accordingly to keep constant row and column sums, and the partial *p*-values are thus summed.

For the lower-tail ('less-than') alternative (`alternative = 2`), the summation starts with *d* as the number of shared members of sets between *i* and *j*, (as with the upper-tail calculation) but then decrements that to 0.

For the 2-tailed alternative, the function sums all the terms with values equal to or less than the the partial *p*-value defined above.

All calculations are done on log-transformed values to avoid numerical underruns:

```
ln(p) = ln(( a + b )!) + ln(( c + d )!) +
        ln(( a + c )!) + ln(( b + d )!) -
        ln(a!) - ln(b!) - ln(c!) - ln(d!) -
        ln(( a + b + c + d )!)
```

Since log-transformed *p*-values cannot be directly added, the so-called log-sum-exponential trick is used to combine them.

See Also

`buildGeneSetNetworkLFFast` `scoreJaccardMatrix_C`

Examples

```
## Not run:
library( GSNA )
LFMatrix <- scoreLFMatrix_C( PresenceAbsMatrix )

## End(Not run)
```

---

scoreOCMatrix_C *scoreOCMatrix_C*

Description

Takes a presence/absence matrix with genes as the rows and modules as columns and calculates a matrix of overlap coefficient values (also known as Szymkiewicz–Simpson coefficients^1).

Usage

```
scoreOCMatrix_C(geneSetCollection_m)
```

Arguments

geneSetCollection_m
: (required) A logical presence/absence matrix representation of a gene set collection in which columns correspond to gene sets, rows correspond to genes and values are TRUE if a gene is present in a gene set and FALSE otherwise. Row and column names correspond to gene symbols and gene set identifiers, respectively. NOTE: for a typical GSNA analysis, this matrix would include only observerved filtered genes and significant gene set hits from pathways analysis. Using a matrix version of the full MSigDB without filtering geneas, for example, would likely be unworkably slow and memory intensive.

Details

The overlap (or Szymkiewicz–Simpson) coefficient for two sets A and B is defined as:

$$OC(A,B) = \frac{|A \cap B|}{\min(|A|, |B|)}$$

Value

This function returns a matrix of overlap coefficient values between gene modules. Values on the diagonal corresponding to self-overlap coefficients are returned as NA.

References

1. M.K V, K K. A Survey on Similarity Measures in Text Mining. MLAIJ. 2016;3: 19–28. doi:10.5121/mlaij.2016.3103

See Also

buildGeneSetNetworkOC scoreLFMatrix_C

Examples

```
## Not run:
library( GSNA )
ocMatrix <- scoreOCMatrix_C( PresenceAbsMatrix )

## End(Not run)
```

`tmod2gsc` *tmod2gsc*

Description

Function takes a tmod or tmodGS object and converts it to a gene set collecton. In the case of a tmod object, the function merely extracts the `$MODULES2GENES` list of character vectors. In the case of tmodGS objects, the list of vectors of numeric gene identifiers in `$gs2gv` is converted to a named list of character vectors of gene names.

Usage

```
tmod2gsc(tmod)
```

Arguments

`tmod`	: a tmod or tmodGS object.

Value

The function returns a gene set collection as a named list of character vectors containing gene names.

See Also

`gsc2tmod()`

`yassifyPathways` *yassifyPathways*

Description

Takes a data.frame and outputs an attractively formatted HTML table widget for reports via the using the DT and DataTables package. Optionally, the user can specify, via the `n` parameter, the number of rows to display in the HTML table. Optionally, IDs in specific columns can be mapped to URLs as links.

Usage

```
yassifyPathways(
  pathways,
  n = NULL,
  url_map_list = list(),
  url_map_by_words_list = list(),
  min_decimal = 5e-04,
  quiet = TRUE,
  table_row_colors = c(`1` = "#EEF", `2` = "#FFD"),
  ...
)
```

Arguments

pathways	A data.frame containing pathways data.
n	(optional) The number of rows that the user wishes to display. This defaults to the total number of rows.
url_map_list	(optional) A list of vectors containing unique IDs and their corresponding URLs as name:value pairs. In the enclosing list, the element names are the names of the columns in the data.frame containing the fields needing to be converted to URL links.
url_map_by_words_list	(optional) Similar to url_map_list, except that instead of mapping the full value of a text field, the function looks for occurrences of key values within the text using gsub() to substitute a URL link tag. This allows fields containing multiple IDs to be converted to a group of URL links.
min_decimal	(optional) The minimal value for decimal format. Below this, scientific notation is used (default 0.0005).
quiet	(optional) If FALSE, this tells the function to emit warnings when an identifier term has no matching URL. By default, this value is TRUE, suppressing this behavior.
table_row_colors	(optional) This argument specifies the row background colors used to contrast different subnets. (default: c("1"="#EEF","2"="#FFD"), pale blue and pale yellow)
...	Additional arguments passed to DT::datatable.

Value

An attractive HTML table widget, optionally with unique IDs represented as links.

Examples

```
## Not run:
   # Export merged pathways/subnets data:
   analysis.mergePathways <- gsnMergePathways( object = analysis.GSN )

   # Convert IDs in analysis.mergePathways into a named list of URLs:
   url_map_l <- list()

   # To link to a gene set page on MSigDB's website, this base URL can be used:
   msig_url <- "http://www.gsea-msigdb.org/gsea/msigdb/geneset_page.jsp"
```

```
# This method works for MSigDB IDs (e.g. "M5928"), with parameter 'systematicName':
url_map_l[['ID']] <-
  with( analysis.mergePathways,
    structure(paste0( msig_url, "?systematicName=", ID),
              names = ID
            )
       )

# If MSigDB STANDARD_NAMES (e.g. "GO_RESPONSE_TO_GLUCAGON") are present in the
# pathways data, they can be linked to URLs using parameter 'geneSetName':
url_map_l[['STANDARD_NAME']] <-
    with( analysis.mergePathways,
      structure( paste0(msig_url, "?geneSetName=", STANDARD_NAME),
                 names = STANDARD_NAME
               )
        )
# Print HTML table:
yassifyPathways( pathways = analysis.mergePathways,
                 n = 200,
                 url_map_list = url_map_l )


## End(Not run)
```

Index