Package 'TMSig'

November 12, 2026

Type Package

Title Tools for Molecular Signatures

Version 1.6.0

Date 2024-10-17

Description The TMSig package contains tools to prepare, analyze, and visualize named lists of sets, with an emphasis on molecular signatures (such as gene or kinase sets). It includes fast, memory efficient functions to construct sparse incidence and similarity matrices and filter, cluster, invert, and decompose sets. Additionally, bubble heatmaps can be created to visualize the results of any differential or molecular signatures analysis.

License GPL (>= 3)

URL https://github.com/EMSL-Computing/TMSig

BugReports https://github.com/EMSL-Computing/TMSig/issues

Encoding UTF-8

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

biocViews Clustering, GeneSetEnrichment, GraphAndNetwork, Pathways, Visualization

Depends R (>= 4.4.0), limma

Imports circlize, ComplexHeatmap, data.table, grDevices, grid, GSEABase, Matrix, methods, stats, utils

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

Config/testthat/edition 3

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

git_branch RELEASE_3_23

git_last_commit 5803a34

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Tyler Sagendorf [aut, cre] (ORCID:
<https://orcid.org/0000-0003-1552-4870>),
Di Wu [ctb],
Gordon Smyth [ctb]

Maintainer Tyler Sagendorf <tyler.sagendorf@pnnl.gov>

Contents

TMSig-package 2
cameraPR.matrix 3
clusterSets 6
decomposeSets 8
enrichmap 10
enrichmapColorFunctions 13
extendRangeNum 14
filterSets 15
incidenceToList 16
invertSets 17
readGMT 18
similarity 19
sparseIncidence 20

Index **23**

`TMSig-package` *TMSig: Tools for Molecular Signatures*

Description

The TMSig package contains tools to prepare, analyze, and visualize named lists of sets, with an emphasis on molecular signatures (such as gene or kinase sets). It includes fast, memory efficient functions to construct sparse incidence and similarity matrices and filter, cluster, invert, and decompose sets. Additionally, bubble heatmaps can be created to visualize the results of any differential or molecular signatures analysis.

Author(s)

Maintainer: Tyler Sagendorf <tyler.sagendorf@pnnl.gov> (ORCID)

Other contributors:

- Di Wu [contributor]
- Gordon Smyth [contributor]

See Also

Useful links:

- https://github.com/EMSL-Computing/TMSig
- Report bugs at https://github.com/EMSL-Computing/TMSig/issues

---

`cameraPR.matrix` *Competitive Gene Set Test Accounting for Inter-gene Correlation*

Description

Pre-ranked Correlation-Adjusted MEan RAnk gene set testing (CAMERA-PR) tests whether a set of genes is highly ranked relative to other genes in terms of some measure of differential expression, accounting for inter-gene correlation (Wu & Smyth, 2012). See `cameraPR` for details.

While the language is gene-centric, any *a priori* groups of molecules may be tested.

Usage

```
## S3 method for class 'matrix'
cameraPR(
  statistic,
  index,
  use.ranks = FALSE,
  inter.gene.cor = 0.01,
  sort = TRUE,
  alternative = c("two.sided", "less", "greater"),
  adjust.globally = FALSE,
  min.size = 2L,
  ...
)
```

Arguments

`statistic`	a matrix of statistics (moderated z-statistics preferred) with genes/molecules as row names and one or more contrasts or coefficients as column names. Missing values are allowed.
`index`	a named list of sets to test. Passed to `sparseIncidence`. `index` must be a list of character vectors, not the result of `ids2indices`, so it is more restrictive than what `cameraPR.default` allows.
`use.ranks`	logical; whether to perform a parametric test (`FALSE`; default) or a rank-based test (`TRUE`).
`inter.gene.cor`	numeric; the inter-gene correlation within tested sets. May be a single value or a named vector with names matching those of `index`.
`sort`	logical; should the results of each contrast be sorted by p-value? Default is `TRUE`.

alternative	character; the alternative hypothesis. Must be one of "two.sided" (default), "greater", or "less". May be abbreviated.
adjust.globally	logical; whether p-values from different contrasts should be adjusted together. It is recommended to set this to TRUE when testing a set of closely related contrasts. See Section 13.3 of the LIMMA User's Guide (limmaUsersGuide) for details. Default is FALSE.
min.size	integer; the minimum set size. To be considered for testing, sets must have at least min.size elements with non-missing values in all contrasts. The default value of 2 is the minimum possible set size required for testing, though a value of 10 or higher is recommended; higher values tend to produce more robust results.
...	other arguments are not currently used

Value

A data.frame with the following columns:

Contrast	factor; the contrast of interest.
GeneSet	character; the gene set being tested.
NGenes	integer; number of genes in the set with values in the statistic matrix for a given contrast.
Correlation	numeric; inter-gene correlation (only included if inter.gene.cor was not a single value).
Direction	character; direction of change ("Up" or "Down").
TwoSampleT	numeric; two-sample t-statistic (only included if use.ranks=FALSE).
df	integer; degrees of freedom (only included if use.ranks=FALSE). Two less than the number of non-missing values in each column of the statistic matrix.
ZScore	numeric; the z-score equivalent of TwoSampleT.
PValue	numeric; one- or two-sided (if alternative="two.sided") p-value.
FDR	numeric; Benjamini and Hochberg FDR adjusted p-value.

Test Assumptions

If use.ranks=FALSE, the parametric version of CAMERA-PR will be used. Since this is a modification of Student's two sample t-test, it is assumed that the statistics in each column of statistic are approximately Normally distributed. In camera.default, the moderated t-statistics are converted to z-statistics with zscoreT and used for the analysis.

If use.ranks=TRUE, a modified Wilcoxon-Mann-Whitney rank sum test will be used.

Author(s)

CAMERA-PR was developed by Di Wu and Gordon Smyth (2012). With permission, Tyler Sagendorf modified their original code to create the matrix method. If using cameraPR.matrix, please cite the original paper, as well as the TMSig R package.

References

Wu, D., and Smyth, G. K. (2012). Camera: a competitive gene set test accounting for inter-gene correlation. *Nucleic Acids Research* 40, e133. doi:10.1093/nar/gks461.

Goeman, J. J., and Bühlmann, P. (2007). Analyzing gene expression data in terms of gene sets: methodological issues. *Bioinformatics* 23, 980-987. doi:10.1093/bioinformatics/btm051.

See Also

cameraPR, rankSumTestWithCorrelation

Examples

```
require(stats)

# Simulate experimental data with control and treatment groups (3 samples
# each)
group <- rep(c("control", "treatment"), each = 3)
design <- model.matrix(~ 0 + group)
contrasts <- makeContrasts(contrasts = "grouptreatment - groupcontrol",
                           levels = colnames(design))

ngenes <- 1000L
nsamples <- length(group)

set.seed(0)
y <- matrix(data = rnorm(ngenes * nsamples),
            nrow = ngenes, ncol = nsamples,
            dimnames = list(paste0("gene", seq_len(ngenes)),
                            make.unique(group)))

# First set of 20 genes are genuinely differentially expressed
# (trt1 and trt2 are lower than control)
index1 <- 1:20
y[index1, 1:3] <- y[index1, 1:3] + 1

# Second set of 20 genes are not DE
index2 <- 21:40

# Generate matrix of moderated t-statistics
fit <- lmFit(y, design)
fit.contr <- contrasts.fit(fit, contrasts = contrasts)
fit.smooth <- eBayes(fit.contr)

index <- list(set1 = rownames(y)[index1],
              set2 = rownames(y)[index2])

# Compute z-score equivalents of moderated t-statistics
statistic <- zscoreT(fit.smooth$t, fit.smooth$df.total)
head(statistic)

# Only set1 is DE
```

```
cameraPR(statistic = statistic, index = index)

# Non-parametric version
cameraPR(statistic = statistic, index = index, use.ranks = TRUE)
```

`clusterSets` *Similarity-Based Clustering of Sets*

Description

Determine clusters of highly similar sets. Used to reduce the redundancy of sets prior to statistical analysis.

Usage

```
clusterSets(
  x,
  type = c("jaccard", "overlap", "otsuka"),
  cutoff = 0.85,
  method = "complete",
  h = 0.9
)
```

Arguments

`x`	a named list of sets. Elements must be of type `"character"`.
`type`	character; the type of similarity measure to use. Either `"jaccard"`, `"overlap"`, or `"otsuka"`. May be abbreviated.
`cutoff`	numeric 0-1; minimum similarity coefficient required to classify two sets as being similar. Default is 0.85.
`method`	character; the clustering method passed to `hclust`. Default is "complete", so sets will only be included in a cluster if their similarity to all other sets in that cluster is $\geq$ `cutoff`.
`h`	numeric 0-1; cut height used to define clusters. Passed to `cutree`. Default is 0.9.

Value

A `data.frame` with 3 columns:

`set`	character; the name of the set.
`cluster`	integer; the cluster identifier.
`set_size`	integer; the size of the set (number of elements).

Results are arranged in ascending order by cluster, descending order by set size, and then alphanumerically by set name.

Function Details

Given a named list of sets, `clusterSets` calculates all pairwise Jaccard, overlap, or Ōtsuka similarity coefficients (see `similarity` for details). Any coefficients below `cutoff` are set to 0 and complete-linkage hierarchical clustering is performed on the dissimilarity matrix (calculated as 1 - coefficients). Lastly, `cutree` is used with cut height `h` to define clusters, and the results are stored in a `data.frame`.

Optimization

Clustering does not need to be performed on those sets that are not sufficiently similar (value of `similarity` below `cutoff`) to any other set, as they will always be placed in their own cluster. By excluding these sets during the hierarchical clustering step, the speed of `clusterSets` will increase as the value of `cutoff` approaches 1 (as the size of the dissimilarity matrix decreases).

Minimum Set Size

Sets that are not sufficiently large will always appear as singleton clusters, unless they are aliased or subsets (overlap similarity only). For two sets A and B to be sufficiently similar, defined as having a similarity coefficient at least equal to some cutoff (e.g., $Jaccard \geq x$), they must have minimum sizes $|A|$, $|B|$, and intersection size $|A \cap B|$:

- **Jaccard:** $|A| = \lceil \frac{x}{1-x} \rceil, \quad |B| = 1 + |A|, \quad |A \cap B| = |A|$
- **Overlap:** $|A| = |B| = 1 + \lceil \frac{x}{1-x} \rceil, \quad |A \cap B| = |A| - 1$
- **Ōtsuka:** $|A| = \lceil \frac{x^2}{1-x^2} \rceil, \quad |B| = 1 + |A|, \quad |A \cap B| = |A|$

where $\lceil y \rceil$ is the ceiling function applied to some real number y.

For example, if the cutoff is $x = 0.85$, then the minimum set and intersection sizes are

- **Jaccard:** $|A| = 6, \quad |B| = 7, \quad |A \cap B| = 6$
- **Overlap:** $|A| = |B| = 7, \quad |A \cap B| = 6$
- **Ōtsuka:** $|A| = 3, \quad |B| = 4, \quad |A \cap B| = 3$

That is, sets with fewer elements or smaller intersections will always appear as singleton clusters unless they are aliased or, in the case of the overlap similarity, subsets.

Source

This function is based on the procedure described in the Molecular Signatures Database (MSigDB) v7.0 Release Notes (Liberzon 2011, 2015): https://docs.gsea-msigdb.org/#MSigDB/Release_Notes/MSigDB_7.0/. Specifically, sections "C2:CP:Reactome — Major Overhaul" and "C5 (Gene Ontology Collection) — Major Overhaul". Though hierarchical clustering is widely used, the defaults are exactly what is set for MSigDB (private correspondence).

References

Liberzon, A., Subramanian, A., Pinchback, R., Thorvaldsdóttir, H., Tamayo, P., & Mesirov, J. P. (2011). Molecular signatures database (MSigDB) 3.0. *Bioinformatics, 27*(12), 1739–1740. doi:10.1093/bioinformatics/btr260

Liberzon, A., Birger, C., Thorvaldsdóttir, H., Ghandi, M., Mesirov, J. P., & Tamayo, P. (2015). The Molecular Signatures Database (MSigDB) hallmark gene set collection. *Cell systems, 1*(6), 417–425. doi:10.1016/j.cels.2015.12.004

See Also

filterSets, similarity

Examples

```
x <- list("A" = letters[1:5],
          "B" = letters[1:4], # subset of A
          "C" = letters[1:4], # aliased with B
          "D" = letters[1:3], # subset of A, B, C
          "E" = c("a", "a", NA), # duplicates and NA
          "F" = c("x", "y", "z"), # distinct elements
          "G" = letters[3:6]) # overlaps with A-E

# Default clustering based on Jaccard similarity
clusterSets(x)
clusterSets(x, type = "overlap") # overlap similarity
clusterSets(x, type = "otsuka") # Otsuka similarity

# Relax Jaccard similarity cutoff
(df <- clusterSets(x, cutoff = 0.5))

# Keep the first (largest) set from each cluster
with(df, set[!duplicated(cluster)]) # A, G, E, F

# Keep the smallest set from each cluster
df <- df[order(df$set_size), ]
with(df, set[!duplicated(cluster)]) # E, D, F, G

# Cluster aliased sets (type = "otsuka" would produce
# identical results)
clusterSets(x, type = "jaccard", cutoff = 1)

# Cluster subsets and aliased sets
clusterSets(x, type = "overlap", cutoff = 1)
```

---

decomposeSets *Decompose Pairs of Overlapping Sets Into 3 Disjoint Parts*

Description

Decompose all pairs of sufficiently overlapping sets into 3 disjoint parts: the elements unique to the first set, the elements unique to the second set, and the elements found in both sets. See the examples section in `invertSets` for a method to decompose an entire list of sets.

Usage

```
decomposeSets(
  x,
  overlap = 1L,
  AND = "~AND~",
  MINUS = "~MINUS~",
  verbose = TRUE
)
```

Arguments

`x`	a named list of sets. Elements must be of type `"character"`.
`overlap`	integer; only pairs of sets with at least `overlap` elements in common will be decomposed.
`AND`	character; string used to denote the intersection of two sets. Defaut is "~AND~", which produces intersections of the form "A ~AND~ B" (i.e., elements in both A and B).
`MINUS`	character; string used to denote the difference of two sets. Defualt is "~MINUS~", which produces differences of the form "A ~MINUS~ B" (i.e., elements in A and not in B).
`verbose`	logical; whether to print warnings and messages.

Value

A named list of disjoint parts of sets. May contain aliases.

Optimization

Since the size of the intersection between two sets is at most the size of the smaller set, any sets with fewer than `overlap` elements can be immediately discarded.

Source

Decomposition of sets is described by Jiang and Gentleman (2007) in section 2.3.1 "Overlap among gene sets". It is a method to reduce the redundancy of significant gene set testing results whereby the decomposed sets are reanalyzed and the following selections can be made:

- If the elements unique to set 1 and set 2, elements common to both sets, or all 3 parts are statistically significant, keep both set 1 and set 2 in the original results. We can not separate their effects.
- If the elements unique to set 1 or the elements unique to set 1 and common to both sets are statistically significant, only keep set 1 in the original results. (The same logic can be applied for set 2.)

References

Jiang, Z., & Gentleman, R. (2007). Extensions to gene set enrichment. *Bioinformatics, 23*(3), 306–313. doi:10.1093/bioinformatics/btl599

See Also

filterSets

Examples

```
x <- list("A" = letters[1:10],
          "B" = letters[3:7],
          "C" = letters[1:4],
          "D" = letters[6:12])

decomposeSets(x)

decomposeSets(x, overlap = 5L)
```

---

enrichmap *Set Enrichment Bubble Heatmap*

Description

Create a bubble heatmap summarizing molecular signature analysis results, such as those from cameraPR.matrix. May also be used to generate bubble heatmaps of differential analysis results.

Usage

```
enrichmap(
  x,
  n_top = 15L,
  set_column = "GeneSet",
  statistic_column = "ZScore",
  contrast_column = "Contrast",
  padj_column = "FDR",
  padj_legend_title = padj_column,
  padj_aggregate_fun = function(padj) median(-log10(padj), na.rm = TRUE),
  padj_cutoff = 0.05,
  plot_sig_only = TRUE,
  padj_fill = "grey",
  colors = c("#3366ff", "darkred"),
  heatmap_color_fun = cameraColorFun,
  scale_by = c("row", "column", "max"),
  cell_size = unit(14, "points"),
  filename,
  height = 5,
```

```
  width = 5,
  units = "in",
  heatmap_args = list(),
  padj_args = list(),
  save_args = list(),
  draw_args = list()
)
```

Arguments

x	an object that can be coerced to a data.table with columns contrast_column, set_column, statistic_column, and padj_column.
n_top	integer; number of sets (rows) to display. Defaults to the top 15 sets with the highest median $-log_{10}$(adjusted p-values) across contrasts.
set_column	character; the name of a column in x containing unique set identifiers that will be used as the row names in the heatmap. Default is "GeneSet".
statistic_column	character; the name of a column in x containing the statistic for each combination of contrast and molecular signature. Determines the heatmap body colors. Default is "ZScore".
contrast_column	character; the name of a column in x containing contrasts that will be used as columns for the heatmap. Entries of x[[rownames_colum]] must be uniquely defined for each contrast group.
padj_column	character; the name of a column in x containing the adjusted p-values. Determines the diameter of each bubble in the heatmap.
padj_legend_title	character; title of the background fill legend. Defaults to padj_column.
padj_aggregate_fun	function; a function used to aggregate the adjusted p-values in x[[pvalue_column]] across contrasts for each unique entry in x[[set_column]]. The default computes the median of the $-log_{10}$ adjusted p-values.
padj_cutoff	numeric; cutoff for terms to be statistically significant. If plot_sig_only=TRUE, only those molecular signatures with at least one padj_column value less than this threshold may appear in the heatmap. Default is 0.05.
plot_sig_only	logical; whether to plot only those n_top terms that have at least one padj_column value less than padj_cutoff.
padj_fill	character; the background color used for values in padj_column that are less than padj_cutoff. Default is "grey".
colors	character; vector of length 2 specifying the colors for the largest negative and largest positive values of x[[statistic_column]], respectively. Default is "#3366ff" (blue) and "darkred".
heatmap_color_fun	function; used to create the legend for the heatmap bubble fill. See enrichmapColorFunctions for details.

scale_by	character; whether to scale the bubbles such that the term with the largest $-log_{10}$ adjusted p-value in each row (scale_by="row"), column (scale_by="column"), or overall (scale_by="max") is of maximum diameter. Default is "row" to better visualize patterns across contrasts. May be abbreviated.
cell_size	unit object; the size of each heatmap cell (used for both height and width). Default is unit(14, "points"). This also controls the default text size, which defaults to 90% the size of cell_size.
filename	character; the file name used to save the heatmap. If missing (default), the heatmap will be displayed instead.
height	numeric; height of the file in units.
width	numeric; width of the file in units.
units	character; units that define height and width. Defaults to "in" (inches). See unit for possible units.
heatmap_args	list; additional arguments passed to Heatmap.
padj_args	list; additional arguments passed to Legend. Modifies the adjusted p-value legend.
save_args	list; additional arguments passed to the graphics device determined by the filename extension. See png and pdf for options.
draw_args	list; additional arguments passed to draw-HeatmapList-method.

Details

The diameter of each bubble is determined by the $-log_{10}$ adjusted p-values. By default, the bubbles are scaled such that the contrast with the largest $-log_{10}$ adjusted p-value per row (scale_by="row") has a bubble diameter of 0.95 * cell_size, and all other bubbles in that row are scaled relative to this maximum diameter; this is to better visualize patterns across contrasts. Bubbles can also be scaled so that largest $-log_{10}$ adjusted p-value by column (scale_by="column") or in the entire heatmap (scale_by="max") has the maximum diameter. The bubble diameters will be no smaller than 0.2 * cell_size.

Value

Nothing. Displays heatmap or saves the heatmap to a file (if filename is provided).

See Also

ComplexHeatmap-package

Examples

```
## Simulate results of cameraPR.matrix
set.seed(1)
df <- 5000L
x <- data.frame(
    Contrast = rep(paste("Contrast", 1:3), each = 4),
    GeneSet = rep(paste("GeneSet", 1:4), times = 3),
    TwoSampleT = 5 * rt(n = 12L, df = df)
```

```
)

# Calculate z-statistics, two-sided p-values, and BH adjusted p-values
x$ZScore <- limma::zscoreT(x = x$TwoSampleT, df = df)
x$PValue <- 2 * pnorm(abs(x$ZScore), lower.tail = FALSE)
x$FDR <- p.adjust(x$PValue, method = "BH")

## Plot results
# Same as enrichmap(x, statistic_column = "ZScore")
enrichmap(x = x,
          set_column = "GeneSet",
          statistic_column = "ZScore",
          contrast_column = "Contrast",
          padj_column = "FDR",
          padj_cutoff = 0.05)

# Include gene sets with adjusted p-values above padj_cutoff (0.05).
# Also update adjusted p-value legend title.
enrichmap(x = x,
          statistic_column = "ZScore",
          plot_sig_only = FALSE,
          padj_legend_title = "BH Adjusted\nP-Value")
```

`enrichmapColorFunctions` *Color Functions for enrichmap*

Description

Heatmap color functions for plotting GSEA-like and CAMERA-like results with `enrichmap`.

Usage

```
gseaColorFun(statistics, colors = c("#3366ff", "darkred"))

cameraColorFun(statistics, colors = c("#3366ff", "darkred"))
```

Arguments

`statistics`	numeric matrix or vector of statistics. Missing values are removed. Used to compute limits for the color legend.
`colors`	vector of 2 colors for the most negative and most positive values of `statistics`. Default is "#3366ff" (blue) and "darkred".

Details

For `gseaColorFun`, the `statistics` are expected to be normalized enrichment scores (NES). Due to how the NES is formulated, values between -1 and 1 are never significant or otherwise interesting, so they are given a white fill so as to not appear in the heatmap (see examples).

Value

a named list of breaks and colors for the heatmap legend.

See Also

extendRangeNum

Examples

```
set.seed(0)
x <- rnorm(10, mean = 0, sd = 2)

cameraColorFun(x)
cameraColorFun(x[x >= 0]) # positive only

gseaColorFun(x)
gseaColorFun(x[x >= 0]) # positive only
```

---

extendRangeNum *Extend the Range of Values Out to the Nearest Digit*

Description

Extend the Range of Values Out to the Nearest Digit

Usage

```
extendRangeNum(x, nearest = 1)
```

Arguments

x	any numeric object. Passed to range.
nearest	numeric; the range of x will be extended out to the value specified by nearest. Default is 1, which extends the range out to the nearest integer.

Value

A numeric vector of length 2 containing the minimum and maximum values of x after extending them outward to the value provided by nearest.

See Also

range, extendrange

Examples

```
set.seed(0)
x <- runif(5, min = -10, max = 10)
range(x) # -4.689827  8.164156

extendRangeNum(x) # -5  9
extendRangeNum(x, nearest = 2) # -6  10
extendRangeNum(x, nearest = 0.1) # -4.7  8.2
```

---

filterSets *Filter a Named List of Sets by Size*

Description

Given a named list of sets, filter to those that contain at least `min_size` and no more than `max_size` elements. The sets are optionally restricted to elements of `background` before filtering by size.

Usage

```
filterSets(x, background = NULL, min_size = 5L, max_size = Inf, ...)

## S4 method for signature 'GeneSet,character,numeric,numeric'
filterSets(x, background = NULL, min_size = 5L, max_size = Inf, ...)

## S4 method for signature 'GeneSetCollection,character,numeric,numeric'
filterSets(x, background = NULL, min_size = 5L, max_size = Inf, ...)
```

Arguments

`x`	a named list of sets. Elements must be of type `"character"`.
`background`	character; optional character vector. `x` will be filtered to only those elements of `background`.
`min_size`	integer (≥ 1); the minimum allowable set size.
`max_size`	integer (≥ 1); the maximum allowable set size.
`...`	additional arguments are not currently used.

Value

A named list of sets at most the same size as `x`.

Examples

```
x <- list("A" = c("a", "b", "c"),
          "B" = c("a", "a", "d", "e", NA), # duplicates and NA
          "C" = c("f", "g"))

# All sets have at least 2 elements,
```

```
# so this just removes duplicates and NA
filterSets(x, min_size = 2L)

# Limit scope of sets before filtering
filterSets(x, min_size = 2L, background = c("a", "c", "e", "z"))
```

`incidenceToList` *Convert Incidence Matrix to a Named List of Sets*

Description

Converts an incidence matrix to a named list of sets. The inverse of `sparseIncidence`.

Usage

```
incidenceToList(incidence)
```

Arguments

`incidence`	incidence matrix with set names as rows and elements as columns. For instance, the output of `sparseIncidence`.

Value

a named list of sets with the same length as `nrow(incidence)`.

Note

Currently, there are no checks to ensure `incidence` is a valid incidence matrix.

Examples

```
x <- list("A" = c("a", "b", "c"),
          "B" = c("c", "d"),
          "C" = c("x", "y", "z", "z"), # duplicates
          "D" = c("a", NA)) # missing values

(imat <- sparseIncidence(x)) # incidence matrix

incidenceToList(incidence = imat)
```

invertSets *Invert a List of Sets, Transposing Sets and Elements*

Description

Invert a list of sets so that elements become set names and set names become elements.

Usage

```
invertSets(x, ...)

## S4 method for signature 'GeneSet'
invertSets(x, ...)

## S4 method for signature 'GeneSetCollection'
invertSets(x, ...)
```

Arguments

x	a named list of sets. Elements must be of type "character".
...	additional arguments are not currently used.

Value

A named list of sets.

Note

This function is essentially a more limited version of purrr::transpose_list.

Examples

```
x <- list("A" = c("a", "b", "c"),
          "B" = c("c", "d"),
          "C" = c("x", "y", "z"),
          "D" = c("a", "c", "d"))

# Invert sets
(y <- invertSets(x))

# Jaccard similarity of pairs of elements
similarity(y)

# Decompose sets into disjoint parts
yc <- lapply(y, paste, collapse = ", ")
invertSets(yc)
```

readGMT *Read GMT File to a Named List of Sets*

Description

Create a named list of sets from a GMT file, or a file structured like a GMT.

Usage

```
readGMT(path, check = TRUE)
```

Arguments

path	character; path to a GMT file. Files may include one additional extension after ".gmt", such as ".gmt.gzip".
check	logical; check that path points to a valid GMT file. If FALSE, files with different extensions may be read, so long as they are in the expected format.

Details

The second entry in each line of the GMT file is assumed to be a URL or some other additional information, so it is discarded.

Value

A named list of character vectors.

Note

Similar to fgsea::gmtPathways.

Examples

```
path <- system.file("extdata", "c5.go.v2023.2.Hs.symbols.gmt.gz",
                    package = "TMSig")

x <- readGMT(path)

head(names(x)) # First 6 gene set names

x[1] # first set
```

`similarity` *Construct a Matrix of Pairwise Set Similarity Coefficients*

Description

Construct a sparse matrix of similarity coefficients for each pair of sets in a list.

Usage

```
similarity(x, type = c("jaccard", "overlap", "otsuka"))
```

Arguments

`x`	a named list of sets. Elements must be of type `"character"`.
`type`	character; the type of similarity measure to use. Either `"jaccard"`, `"overlap"`, or `"otsuka"`. May be abbreviated.

Value

A symmetric `dgCMatrix` containing all pairwise set similarity coefficients.

Set Similarity

If A and B are sets, we define the Jaccard similarity coefficient J as the size of their intersection divided by the size of their union (Jaccard, 1912):

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

The overlap coefficient is defined as the size of the intersection divided by the size of the smaller set (Simpson, 1943, 1947, 1960; Fallaw 1979):

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

The Ōtsuka coefficient is defined as the size of the intersection divided by the geometric mean of the set sizes (Ōtsuka, 1936), which is equivalent to the cosine similarity of two bit vectors:

$$\bar{\mathrm{O}}\mathrm{tsuka}(A, B) = \frac{|A \cap B|}{\sqrt{|A| \times |B|}}$$

The Jaccard and Ōtsuka coefficients can identify aliased sets (sets which contain the same elements, but have different names), while the overlap coefficient can identify both aliased sets and subsets. Aliases and subsets are not easily distinguished without also having the matrix of Jaccard (or Ōtsuka) coefficients or the set sizes.

Notice the relationship between the similarity coefficients:

$$0 \leq \mathrm{J}(A, B) \leq \bar{\mathrm{O}}\mathrm{tsuka}(A, B) \leq \mathrm{Overlap}(A, B) \leq 1$$

Optimization

Calculations are only performed for pairs of sets with nonzero intersections in the lower triangular part of the matrix. As such, `similarity` is efficient even for large similarity matrices, and it is especially efficient for sparse similarity matrices.

References

Jaccard, P. (1912). The distribution of the flora in the alpine zone. *The New Phytologist, 11*(2), 37–50. doi:10.1111/j.1469-8137.1912.tb05611.x. https://www.jstor.org/stable/2427226

Ōtsuka, Y. (1936). The faunal character of the Japanese Pleistocene marine Mollusca, as evidence of the climate having become colder during the Pleistocene in Japan. *Bulletin of the Biogeographical Society of Japan, 6*(16), 165–170.

Simpson, G. G. (1943). Mammals and the nature of continents. *American Journal of Science, 241*(1), 1–31.

Simpson, G. G. (1947). Holarctic mammalian faunas and continental relationships during the Cenozoic. *Bulletin of the Geological Society of America, 58*(7), 613–688.

Simpson, G. G. (1960). Notes on the measurement of faunal resemblance. *American Journal of Science, 258-A*, 300–311.

Fallaw, W. C. (1979). A test of the Simpson coefficient and other binary coefficients of faunal similarity. *Journal of Paleontology, 53*(4), 1029–1034. http://www.jstor.org/stable/1304126

See Also

`sparseIncidence`, `clusterSets`

Examples

```
x <- list("A" = c("a", "b", "c", "d", "e"),
          "B" = c("d", "e", "f", "g"), # overlaps with A
          "C" = c("d", "e", "f", "g"), # aliased with B
          "D" = c("a", "b", "c")) # subset of A

similarity(x) # Jaccard coefficients

similarity(x, type = "overlap") # overlap coefficients

similarity(x, type = "otsuka") # Ōtsuka coefficients
```

---

sparseIncidence *Construct a Sparse Incidence Matrix*

Description

Construct a sparse incidence matrix from a named list of sets.

Usage

```
sparseIncidence(x, ...)

## S4 method for signature 'GeneSet'
sparseIncidence(x, ...)

## S4 method for signature 'GeneSetCollection'
sparseIncidence(x, ...)
```

Arguments

x	a named list of sets. Elements must be of type "character".
...	additional arguments are not currently used.

Details

sparseIncidence differs from GSEABase::incidence in that it returns a sparse matrix, rather than a dense matrix, so it is more memory efficient. It also removes missing elements, removes empty sets, and combines sets with duplicate names.

Value

An object of class dgCMatrix with unique set names as rows and unique elements as columns.

Incidence Matrix

An incidence matrix, A, is defined such that

$$A_{ij} = \begin{cases} 1, & \text{if element } e_j \in \text{set } s_i \\ 0, & \text{otherwise} \end{cases}$$

See Also

incidenceToList, similarity, sparseMatrix

Examples

```
x <- list("A" = c("a", "b"),
          "A" = c("c"), # duplicate sets
          "B" = c("c", "d"),
          "C" = c("x", "y", "z", "z"), # duplicates
          "D" = c("a", NA)) # missing values

(imat <- sparseIncidence(x))

# Sizes of sets and their pairwise intersections
tcrossprod(imat)

# Number of sets in which each element and pair of elements appears
```

```
crossprod(imat)

# Count number of elements unique to each set
keep <- apply(imat, 2, sum) == 1
apply(imat[, keep], 1, sum)
```

Index

∗ **internal**
    TMSig-package, 2

camera.default, *4*
cameraColorFun
        *(*enrichmapColorFunctions*)*, 13
cameraPR, *3*, *5*
cameraPR.default, *3*
cameraPR.matrix, 3, *10*
clusterSets, 6, *20*
cutree, *6, 7*

decomposeSets, 8
dgCMatrix, *19*, *21*

enrichmap, 10, *13*
enrichmapColorFunctions, *11*, 13
extendrange, *14*
extendRangeNum, *14*, 14

filterSets, *8*, *10*, 15
filterSets,GeneSet,character,numeric,numeric-method
        *(*filterSets*)*, 15
filterSets,GeneSetCollection,character,numeric,numeric-method
        *(*filterSets*)*, 15

GSEABase::incidence, *21*
gseaColorFun *(*enrichmapColorFunctions*)*,
        13

hclust, *6*
Heatmap, *12*

ids2indices, *3*
incidenceToList, 16, *21*
invertSets, *9*, 17
invertSets,GeneSet-method *(*invertSets*)*,
        17
invertSets,GeneSetCollection-method
        *(*invertSets*)*, 17

Legend, *12*
limmaUsersGuide, *4*

numeric, *14*

pdf, *12*
png, *12*

range, *14*
rankSumTestWithCorrelation, *5*
readGMT, 18

similarity, *7, 8*, 19, *21*
sparseIncidence, *3*, *16*, *20*, 20
sparseIncidence,GeneSet-method
        *(*sparseIncidence*)*, 20
sparseIncidence,GeneSetCollection-method
        *(*sparseIncidence*)*, 20
sparseMatrix, *21*

TMSig *(*TMSig-package*)*, 2
TMSig-package, 2

unit, *12*

zscoreT, *4*