Package: GeDi (via r-universe)

October 31, 2026

Title Defining and visualizing the distances between different genesets

Version 1.8.0

Date 2025-11-26

Description The package provides different distances measurements to calculate the difference between genesets. Based on these scores the genesets are clustered and visualized as graph. This is all presented in an interactive Shiny application for easy usage.

Depends R (>= 4.4.0)

Imports Matrix, shiny, shinyWidgets, bs4Dash, rintrojs, utils, DT, dplyr, shinyBS, STRINGdb, igraph, visNetwork, shinycssloaders, fontawesome, grDevices, parallel, stats, ggplot2, plotly, expm, RColorBrewer, scales, readxl, ggdendro, ComplexHeatmap, BiocNeighbors, tm, wordcloud2, tools, BiocParallel, BiocFileCache, cluster, methods, circlize, proxyC, simona

Suggests knitr, rmarkdown, testthat (>= 3.0.0), DESeq2, mosdef, GeneTonic, htmltools, AnnotationDbi, macrophage, topGO, biomaRt, ReactomePA, clusterProfiler, BiocStyle, org.Hs.eg.db

License MIT + file LICENSE

Encoding UTF-8

VignetteBuilder knitr

URL https://github.com/AnnekathrinSilvia/GeDi

BugReports https://github.com/AnnekathrinSilvia/GeDi/issues

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

biocViews GUI, GeneSetEnrichment, Software, Transcription, RNASeq, Visualization, Clustering, Pathways, ReportWriting, GO, KEGG, Reactome, ShinyApps

Config/pak/sysreqs
cmake libglpk-dev make libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev perl zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 13:02:56 UTC

RemoteUrl https://github.com/bioc/GeDi

RemoteRef RELEASE_3_23

RemoteSha 28cf7810d7ec9027791a67838b985de9c2e3e567

Contents

.checkGenesets . . . 3
.checkGTL . . . 4
.checkPPI . . . 4
.checkScores . . . 5
.cluster_markov . . . 5
.filterGenesets . . . 7
.findSeparator . . . 7
.getClusterDatatable . . . 8
.getGenesetDescriptions . . . 8
.getNumberCores . . . 9
.graphMetricsGenesetsDT . . . 9
.map_to_color . . . 10
.sepguesser . . . 11
buildClusterGraph . . . 11
buildGraph . . . 12
buildHistogramData . . . 13
checkInclusion . . . 14
clustering . . . 15
deprecated . . . 16
distanceDendro . . . 17
distanceHeatmap . . . 17
enrichmentWordcloud . . . 19
fuzzyClustering . . . 20
GeDi . . . 21
getAdjacencyMatrix . . . 23
getAnnotation . . . 23
getBipartiteGraph . . . 24
getClusterAdjacencyMatrix . . . 25
getGenes . . . 25
getGraphTitle . . . 26
getId . . . 27
getInteractionScore . . . 28
getJaccardMatrix . . . 29
getKappaMatrix . . . 30
getMeetMinMatrix . . . 30
getpMMMatrix . . . 31

getPPI 32
getSorensenDiceMatrix 33
getStringDB 34
goDistance 35
gsHistogram 36
kMeansClustering 37
kNN_clustering 38
louvainClustering 38
macrophage_KEGG_example 39
macrophage_Reactome_example 40
macrophage_topGO_example 40
macrophage_topGO_example_small 41
markovClustering 41
pamClustering 42
path_to_GeDi 43
pMMlocal 44
ppi_macrophage_topGO_example_small 45
prepareGenesetData 45
sample_geneset 46
sample_geneset_broken 47
sample_geneset_empty 47
sample_geneset_small 48
scores_macrophage_topGO_example_small 48
seedFinding 49

Index **50**

`.checkGenesets` *Check genesets format*

Description

Check if the input genesets have the expected format for this app

Usage

```
.checkGenesets(
  genesets,
  col_name_genesets = "Genesets",
  col_name_genes = "Genes"
)
```

Arguments

`genesets`	a `list`, A `list` of genesets where each genesets is represented by `list` of genes.
`col_name_genesets`	character, the name of the column in which the geneset ids are listed. Defaults to "Genesets".

col_name_genes character, the name of the column in which the genes are listed. Defaults to "Genes".

Value

A validated and formatted genesets data frame.

.checkGTL *Check GeneTonic List format*

Description

Check if the provided GeneTonic List object has the expected format for the app and extract the functional enrichment results

Usage

```
.checkGTL(gtl)
```

Arguments

gtl A GeneTonicListobject generated with GeneTonic::GeneTonic_list(), containing the functional enrichment results.

Value

A validated and renamed geneset data.frame.

.checkPPI *Check PPI format*

Description

Check if the Protein-Protein-interaction (PPI) has the expected format for this app

Usage

```
.checkPPI(ppi)
```

Arguments

ppi a data.frame, Protein-protein interaction (PPI) network data frame. The object is expected to have three columns, Gene1 and Gene2 which specify the gene names of the interacting proteins in no particular order (symmetric interaction) and a column combined_score which is a numerical value of the strength of the interaction.

Value

A validated and formatted PPI data frame.

.checkScores *Check distance scores format*

Description

Check if the provided distance scores have the expected format for this app

Usage

```
.checkScores(genesets, distance_scores)
```

Arguments

genesets	a list, A list of genesets where each genesets is represented by list of genes.
distance_scores	A Matrix::Matrix() or object, A matrix with numerical (distance) scores.

Value

A validated and formatted distance_scores Matrix::Matrix().

.cluster_markov *Markov Clustering (MCL) for community detection*

Description

This function implements the Markov Clustering (MCL) algorithm for finding community structure, in an analogous way to other existing algorithms in igraph.

Usage

```
.cluster_markov(
  g,
  add_self_loops = TRUE,
  loop_value = 1,
  mcl_expansion = 2,
  mcl_inflation = 2,
  allow_singletons = TRUE,
  max_iter = 100,
  return_node_names = TRUE,
  return_esm = FALSE
)
```

Arguments

g	The input graph object
add_self_loops	Logical, whether to add self-loops to the matrix by setting the diagonal to loop_value
loop_value	Numeric, the value to use for self-loops
mcl_expansion	Numeric, cluster expansion factor for the Markov clustering iteration - defaults to 2
mcl_inflation	Numeric, cluster inflation factor for the Markov clustering iteration - defaults to 2
allow_singletons	Logical; if TRUE, single isolated vertices are allowed to form their own cluster. If set to FALSE, all clusters of size = 1 are grouped in one cluster (to be interpreted as background noise).
max_iter	Numeric value for the maximum number of iterations for the Markov clustering
return_node_names	Logical, if the graph is named and set to TRUE, returns the node names.
return_esm	Logical, controlling whether the equilibrium state matrix should be returned

Details

This implementation has been driven by the nice explanations provided in

- https://sites.cs.ucsb.edu/~xyan/classes/CS595D-2009winter/MCL_Presentation2.pdf
- https://medium.com/analytics-vidhya/demystifying-markov-clustering-aeb6cdabbfc7
- https://github.com/GuyAllard/markov_clustering (python implementation)

More info on the MCL: https://micans.org/mcl/index.html, and https://micans.org/mcl/sec_description1.html

Value

This function returns a communities object, containing the numbers of the assigned membership (in the slot membership). Please see the igraph::communities() manual page for additional details

References

van Dongen, S.M., Graph clustering by flow simulation (2000) PhD thesis, Utrecht University Repository - https://dspace.library.uu.nl/handle/1874/848

Enright AJ, van Dongen SM, Ouzounis CA, An efficient algorithm for large-scale detection of protein families (2002) Nucleic Acids Research, Volume 30, Issue 7, 1 April 2002, Pages 1575–1584, https://doi.org/10.1093/nar/30.7.1575

Examples

```
library("igraph")
g <- make_full_graph(5) %du% make_full_graph(5) %du% make_full_graph(5)
g <- add_edges(g, c(1, 6, 1, 11, 6, 11))
.cluster_markov(g)
V(g)$color <- .cluster_markov(g)$membership
plot(g)
```

.filterGenesets *Filter Genesets from the input data*

Description

Filter a preselected list of genesets from a `data.frame` of genesets

Usage

```
.filterGenesets(remove, df_genesets)
```

Arguments

`remove`	a `list`, A list of geneset names to be removed
`df_genesets`	a `data.frame`, A `data.frame` with at least two columns. One should be called `Geneset`, containing the names/identifiers of the genesets in the data. The second column should be called `Genes` and contains one string of the genes contained in each geneset.

Value

A `data.frame` containing information about filtered genesets

.findSeparator *Make an educated guess on the separator character*

Description

This function tries to guess which separator was used in a list of delimited strings.

Usage

```
.findSeparator(stringList, sepList = c(",", "\t", ";", " ", "/"))
```

Arguments

`stringList`	`list`, a `list` of strings
`sepList`	`list`, containing the candidates for being identified as separators. Defaults to `c(",", "\t", ";"," ", "/")`.

Value

character, corresponding to the guessed separator. One of "," (comma), "\t" (tab), ";" (semicolon)," " (whitespace) or "/" (backslash).

References

See https://github.com/federicomarini/ideal for details on the original implementation.

.getClusterDatatable *Map each geneset to the cluster it belongs*

Description

Map each geneset to the cluster it belongs and return the information as a `data.frame`

Usage

```
.getClusterDatatable(cluster, gs_names, gs_description)
```

Arguments

`cluster`	A `list` of clusters
`gs_names`	A vector of geneset names
`gs_description`	A vector of descriptions for each geneset

Value

A `data.frame` mapping each geneset to the cluster(s) it belongs to

.getGenesetDescriptions *Get gene set descriptions*

Description

Extracts gene set descriptions from a provided gene set object. The function prioritizes columns "Term", "Description", or "Genesets" to find the appropriate descriptions. If any descriptions are duplicated, the function appends a suffix to make them unique.

Usage

```
.getGenesetDescriptions(genesets)
```

Arguments

`genesets`	a `data.frame`, A `data.frame` with at least two columns. One should be called `Geneset`, containing the names/identifiers of the genesets in the data. The second column should be called `Genes` and contains one string of the genes contained in each geneset.

Value

a `list` of geneset descriptions

.getNumberCores *Determine the number of cores to use for a function*

Description

Determine the number of CPU cores the scoring functions should use when computing the distance scores.

Usage

```
.getNumberCores(n_cores = NULL)
```

Arguments

n_cores	numeric, number of cores to use for the function. Defaults to Null in which case the function takes half of the available cores.

Value

Number of CPU cores to be used.

.graphMetricsGenesetsDT *Generate a* data.frame *of graph metrics*

Description

Generate a data.frame of the graph metrics degree, betweenness, harmonic centrality and clustering coefficient for each node in a given graph.

Usage

```
.graphMetricsGenesetsDT(g, genesets)
```

Arguments

g	A igraph graph object
genesets	A data.frame of genesets with a column Genesets containing geneset identifiers and a column Genes containing the genes belonging to each geneset

Value

A data.frame of geneset extended by columns for the degree, betweenness, harmonic centrality and clustering coefficient for each geneset.

`.map_to_color` *Maps numeric values to color values*

Description

Maps numeric continuous values to values in a color palette

Usage

```
.map_to_color(x, pal, symmetric = TRUE, limits = NULL)
```

Arguments

`x`	A character vector of numeric values (e.g. log2FoldChange values) to be converted to a vector of colors
`pal`	A vector of characters specifying the definition of colors for the palette, e.g. obtained via `RColorBrewer::brewer.pal()`
`symmetric`	Logical value, whether to return a palette which is symmetrical with respect to the minimum and maximum values - "respecting" the zero. Defaults to `TRUE`.
`limits`	A vector containing the limits of the values to be mapped. If not specified, defaults to the range of values in the `x` vector.

Value

A vector of colors, each corresponding to an element in the original vector

Examples

```
a <- 1:9
pal <- RColorBrewer::brewer.pal(9, "Set1")
.map_to_color(a, pal)
plot(a, col = .map_to_color(a, pal), pch = 20, cex = 4)

b <- 1:50
pal2 <- grDevices::colorRampPalette(
  RColorBrewer::brewer.pal(name = "RdYlBu", 11)
)(50)
plot(b, col = .map_to_color(b, pal2), pch = 20, cex = 3)
```

.sepguesser *Make an educated guess on the separator character*

Description

This function tries to guess which separator was used in a text delimited file.

Usage

```
.sepguesser(file, sep_list = c(",", "\t", ";", " ", "/"))
```

Arguments

file	a character, location of a file to read data from.
sep_list	a list, containing the candidates for being identified as separators. Defaults to c(",", "\t", ";"," ", "/").

Value

A character, corresponding to the guessed separator. One of "," (comma), "\t" (tab), ";" (semi-colon)," " (whitespace) or "/" (backslash).

References

See https://github.com/federicomarini/ideal for details on the original implementation.

buildClusterGraph *Build a cluster graph*

Description

Build a igraph from cluster information, connecting nodes which belong to the same cluster.

Usage

```
buildClusterGraph(
  cluster,
  geneset_df,
  gs_ids,
  color_by = NULL,
  gs_names = NULL
)
```

Arguments

cluster	list, a list of clusters, where each cluster member is indicated by a numeric value.
geneset_df	data.frame, a data.frame of genesets with at least two columns, one called Genesets containing geneset identifiers and one called Genes containing a list of genes belonging to the individual genesets.
gs_ids	vector, a vector of geneset identifiers, e.g. the Genesets column of geneset_df.
color_by	character, a column name of geneset_df which is used to color the nodes of the resulting graph. The column should ideally contain a numeric measurement. Defaults to NULL and nodes will remain uncolored.
gs_names	vector, a vector of geneset descriptions/names, e.g. the Term / Description column of geneset_df.

Value

An igraph object to be further manipulated or processed/plotted (e.g. via igraph::plot.igraph() or visNetwork::visIgraph())

Examples

```
cluster <- list(c(1:5), c(6:9, 1))
genes <- list(
  c("PDHB", "VARS2"), c("IARS2", "PDHA1"),
  c("AAAS", "ABCE1"), c("ABI1", "AAR2"), c("AATF", "AMFR"),
  c("BMS1", "DAP3"), c("AURKAIP1", "CHCHD1"), c("IARS2"),
  c("AHI1", "ALMS1")
)
gs_names <- c("a", "b", "c", "d", "e", "f", "g", "h", "i")
gs_ids <- c(1:9)
geneset_df <- data.frame(
  Genesets = gs_names,
  value = rep(1, 9)
)
geneset_df$Genes <- genes
graph <- buildClusterGraph(
  cluster = cluster,
  geneset_df = geneset_df,
  gs_ids = gs_ids,
  color_by = "value",
  gs_names = gs_names
)
```

---

buildGraph *Construct a graph*

Description

Construct a graph from a given adjacency matrix

Usage

```
buildGraph(adjMatrix, geneset_df = NULL, gs_names = NULL, weighted = FALSE)
```

Arguments

adjMatrix	A Matrix::Matrix() indicating for which pair of nodes an edge should be added; 1 indicating an edge, 0 indicating no edge.
geneset_df	data.frame, a data.frame of genesets with at least two columns, one called Genesets containing geneset identifiers and one called Genes containing a list of genes belonging to the individual genesets.
gs_names	vector, a vector of geneset descriptions/names, e.g. the Term / Description column of geneset_df.
weighted	logical value, whether or not the resulting graph should have weighted edges. If TRUE, the adjMatrix values will be used as weights. Default to FALSE.

Value

An igraph object to be further manipulated or processed/plotted (e.g. via igraph::plot.igraph() or visNetwork::visIgraph())

Examples

```
adj <- Matrix::Matrix(0, 100, 100)
adj[c(80:100), c(80:100)] <- 1
geneset_names <- as.character(stats::runif(100, min = 0, max = 1))
rownames(adj) <- colnames(adj) <- geneset_names
graph <- buildGraph(adj)
```

---

buildHistogramData *Prepare data for* gsHistogram().

Description

Prepare the data for the gsHistogram() by generating a data.frame which maps geneset names / identifiers to the size of their size.

Usage

```
buildHistogramData(
  genesets,
  gs_names,
  gs_description = NULL,
  start = 0,
  end = 0
)
```

Arguments

genesets	a list, A list of genesets where each genesets is represented by list of genes.
gs_names	character vector, Name / identifier of the genesets in genesets
gs_description	Optional, a character vector containing a short description for each geneset
start	numeric, Optional, describes the minimum gene set size to include. Defaults to 0.
end	numeric, Optional, describes the maximum gene set size to include. Defaults to 0.

Value

A data.frame mapping geneset names to sizes

Examples

```
## Mock example showing how the data should look like
gs_names <- c("a", "b", "c", "d", "e", "f", "g", "h", "i")
genesets <- list(
  c("PDHB", "VARS2"), c("IARS2", "PDHA1"),
  c("AAAS", "ABCE1"), c("ABI1", "AAR2"), c("AATF", "AMFR"),
  c("BMS1", "DAP3"), c("AURKAIP1", "CHCHD1"), c("IARS2"),
  c("AHI1", "ALMS1")
)

p <- buildHistogramData(genesets, gs_names)

## Example using the data available in the package
data(macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())
genes <- GeDi::prepareGenesetData(macrophage_topGO_example_small)
p <- buildHistogramData(genes, macrophage_topGO_example_small$Genesets)
```

---

checkInclusion *Check for subset inclusion*

Description

Remove subsets from a given list of sets, i.e. remove sets which are completely contained in any other larger set in the list.

Usage

```
checkInclusion(seeds)
```

Arguments

seeds	A list of sets

Value

A `list` of unique sets

Examples

```
## Mock example showing how the data should look like

seeds <- list(c(1:5), c(2:5), c(6:10))
s <- checkInclusion(seeds)

## Example using the data available in the package
data(scores_macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())

seeds <- seedFinding(scores_macrophage_topGO_example_small,
                     simThreshold = 0.3,
                     memThreshold = 0.5)
seeds <- checkInclusion(seeds)
```

`clustering` *Cluster genesets.*

Description

This function performs clustering on a set of scores using either the Louvain or Markov method.

Usage

```
clustering(scores, threshold, cluster_method = "louvain")
```

Arguments

`scores`	A `Matrix::Matrix()` of (distance) scores
`threshold`	numerical, A threshold used to determine which genesets are considered similar. Genesets are considered similar if (distance) score <= threshold. similar.
`cluster_method`	character, the clustering method to use. The options are `louvain` and `markov`. Defaults to `louvain`.

Value

A `list` of clusters

Examples

```
## Mock example showing how the data should look like
m <- Matrix::Matrix(stats::runif(100, min = 0, max = 1), 10, 10)
rownames(m) <- colnames(m) <- c("a", "b", "c", "d", "e",
                                "f", "g", "h", "i", "j")
cluster <- clustering(m, 0.3, "markov")

## Example using the data available in the package
data(scores_macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())

clustering <- clustering(scores_macrophage_topGO_example_small,
                         threshold = 0.5)
```

`deprecated` *Deprecated functions in GeDi*

Description

Functions that are on their way to the function afterlife. Their successors are also listed.

Arguments

`...`	Ignored arguments.

Details

The successors of these functions are likely coming from a renaming of the functions to more intuitive function names

Value

All functions throw a warning, with a deprecation message pointing towards its descendent (if available).

Renaming funciton with more intuitive names

- `getGenes()`, now replaced by the more intuitive name `prepareGenesetData()`. The only change in its functionality concerns the function name.

Author(s)

Annekathrin Nedwed

Examples

```
# try(getGenes())
```

distanceDendro *Plot a dendrogram*

Description

Plot a dendrogram of a matrix of (distance) scores.

Usage

```
distanceDendro(distance_scores, cluster_method = "average")
```

Arguments

distance_scores	A Matrix::Matrix() containing (distance) scores between 0 and 1.
cluster_method	character, indicating the clustering method for the stats::hclust() function. See the stats::hclust() function for the available options. Defaults to 'average'.

Value

A ggdendro::ggdendrogram() plot object.

Examples

```
## Mock example showing how the data should look like

distance_scores <- Matrix::Matrix(0.5, 20, 20)
distance_scores[c(11:15), c(2:6)] <- 0.2
dendro <- distanceDendro(distance_scores, cluster_method = "single")

## Example using the data available in the package
data(scores_macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())
dendro <- distanceDendro(scores_macrophage_topGO_example_small,
                         cluster_method = "average")
```

distanceHeatmap *Plot a heatmap*

Description

Plot a heatmap of a matrix of (distance) scores of the input genesets

Usage

```
distanceHeatmap(
  distance_scores,
  chars_limit = 50,
  plot_labels = TRUE,
  cluster_rows = TRUE,
  cluster_columns = TRUE,
  title = "Distance Scores",
  display_similarity = FALSE,
  quantile_limits = c(0.025, 0.975)
)
```

Arguments

distance_scores	A Matrix::Matrix() of (distance) scores for each pairwise combination of genesets.
chars_limit	Numeric value, Indicates how many characters of the row and column names of distance_scores should be plotted. Defaults to 50 and prevents crowded axes due to long names.
plot_labels	Logical, Indicates if row and collabels should be plotted. Defaults to TRUE
cluster_rows	Logical, Indicates whether or not the rows should be clustered based on the distance scores. Defaults to TRUE
cluster_columns	Logical, Indicates whether or not the rows should be clustered based on the distance scores. Defaults to TRUE
title	character, a title for the figure. Defaults to "Distance Scores"
display_similarity	Logical, Indicates whether or not the scores should be plotted as a distance matrix (i.e. 0 indicates completely identical sets) or as a similarity matrix (i.e. 1 indicates completely identical sets). Defaults to FALSE (i.e. a distance matrix).
quantile_limits	Numerical vector, Used to scale the colors in the heatmap between the given quantiles. Can be helpful in order to reduce the effect of the diagonal or other outlier values on the color scheme. Defaults to c(0.025, 0.975).

Value

A ComplexHeatmap::Heatmap() plot object.

Examples

```
## Mock example showing how the data should look like

distance_scores <- Matrix::Matrix(0.5, 20, 20)
distance_scores[c(11:15), c(2:6)] <- 0.2
rownames(distance_scores) <- colnames(distance_scores) <- as.character(c(1:20))
p <- distanceHeatmap(distance_scores)
```

```
## Example using the data available in the package
data(scores_macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())
p <- distanceHeatmap(scores_macrophage_topGO_example_small)
```

`enrichmentWordcloud` *Visualize the results of an enrichment analysis as word cloud*

Description

Visualize the results of an enrichment analysis as a word cloud. The word cloud highlights the most frequent terms associated with the description of the genesets in the enrichment analysis.

Usage

```
enrichmentWordcloud(
  genesets_df,
  remove_generic_terms = FALSE,
  terms_to_remove = c()
)
```

Arguments

`genesets_df`	A `data.frame` object of an enrichment analysis results. This object should follow the input requirements of `GeDi()`, check out the vignette for further details. Besides the specified required columns, the object should ideally include a column with a short geneset description which is used for the word cloud. If no such column is available, the row names of the `data.frame` are used for the word cloud.
`remove_generic_terms`	Logical, If generic terms like "via", "protein", "factor", "side", "type", "specific", "regulation" and "process" should be removed from the wordcloud. Default to FALSE.
`terms_to_remove`	Character vector, A vector of additional terms that should be removed from the wordcloud.

Value

A `wordcloud2::wordcloud2()` plot object

Examples

```
## Mock example showing how the data should look like

## If no "Term" or "Description" column is available,
## the rownames of the data frame will be used.
geneset_df <- data.frame(
              Genesets = c("GO:0002503", "GO:0045087", "GO:0019886"),
              Genes = c("B2M, HLA-DMA, HLA-DMB",
                        "ACOD1, ADAM8, AIM2",
                        "B2M, CD74, CTSS")
)
rownames(geneset_df) <- geneset_df$Genesets

wordcloud <- enrichmentWordcloud(geneset_df)

## With available "Term" column.
geneset_df <- data.frame(
              Genesets = c("GO:0002503", "GO:0045087", "GO:0019886"),
              Genes = c("B2M, HLA-DMA, HLA-DMB",
                        "ACOD1, ADAM8, AIM2",
                        "B2M, CD74, CTSS"),
              Term = c(
              "peptide antigen assembly with MHC class II protein complex",
              "innate immune response",
              "antigen processing and presentation of exogenous
               peptide antigen via MHC class II")
)

wordcloud <- enrichmentWordcloud(geneset_df)


## Example using the data available in the package

data(macrophage_topGO_example,
     package = "GeDi",
     envir = environment())
wordcloud <- enrichmentWordcloud(macrophage_topGO_example)
```

`fuzzyClustering` *Find cluster from initial seeds*

Description

Merge the initially determined seeds to clusters.

Usage

```
fuzzyClustering(seeds, threshold)
```

Arguments

seeds	A list of seeds, e.g. determined by GeDi::seedFinding() function
threshold	numerical, A threshold for merging seeds

Value

A list of clusters

References

See https://david.ncifcrf.gov/helps/functional_classification.html#clustering for details on the original implementation

Examples

```
## Mock example showing how the data should look like

seeds <- list(c(1:5), c(6:10))
cluster <- fuzzyClustering(seeds, 0.5)

## Example using the data available in the package
data(scores_macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())

seeds <- seedFinding(scores_macrophage_topGO_example_small,
                     simThreshold = 0.3,
                     memThreshold = 0.5)
cluster <- fuzzyClustering(seeds, threshold = 0.5)
```

GeDi *GeDi main function*

Description

GeDi main function

Usage

```
GeDi(
  genesets = NULL,
  ppi_df = NULL,
  distance_scores = NULL,
  gtl = NULL,
  col_name_genesets = "Genesets",
  col_name_genes = "Genes"
)
```

Arguments

genesets	a data.frame, The input data used for GeDi. This should be a data.frame of at least two columns.One column should be called "Genesets" and contain some sort of identifiers for the individual genesets. In this application, we use the term "Genesets" to refer to collections of individual genes, which share common biological characteristics or functions. Such genesets can for example be obtained from databases such as the Gene Ontology (GO), the Kyoto Encyclopedia of Genes and Genomes (KEGG), Reactome, or the Molecular Signatures Database (MSigDB). The identifiers used in these databases can be directly used as geneset identifiers in GeDi. The second column should be called "Genes" and contain a list of genes belonging to the individual genesets in the "Genesets" column. In order to leverage all of the functionality available in GeDi, the column has to contain gene names and no other commonly used identifiers. The column names are case sensitive.
ppi_df	a data.frame, Protein-protein interaction (PPI) network data frame. The object is expected to have three columns, Gene1 and Gene2 which specify the gene names of the interacting proteins in no particular order (symmetric interaction) and a column combined_score which is a numerical value of the strength of the interaction.
distance_scores	A Matrix::Matrix() of (distance) scores
gtl	A GeneTonicListobject generated with GeneTonic::GeneTonic_list(), containing the functional enrichment results.
col_name_genesets	character, the name of the column in which the geneset ids are listed. Defaults to "Genesets".
col_name_genes	character, the name of the column in which the genes are listed. Defaults to "Genes".

Value

A Shiny app object is returned

Examples

```
if (interactive()) {
  GeDi()
}
# Alternatively, you can also start the application with your data directly
# loaded.

data("macrophage_topGO_example", package = "GeDi")
if (interactive()) {
  GeDi(genesets = macrophage_topGO_example)
}
```

getAdjacencyMatrix *Construct an adjacency matrix*

Description

Construct an adjacency matrix from the (distance) scores and a given threshold.

Usage

```
getAdjacencyMatrix(distanceMatrix, cutOff, weighted = FALSE)
```

Arguments

distanceMatrix	A Matrix::Matrix() containing (distance) scores between 0 and 1.
cutOff	Numeric value, indicating for which pair of entries in the distanceMatrix a 1 should be inserted in the adjacency matrix. A 1 is inserted when for each entry in the matrix that is smaller or equal to the cutOff value.
weighted	logical value, indicating whether or not the resulting adjacency matrix should be weighted. If TRUE, the matrix will be weighted by the distance scores in distanceMatrix. Defaults to FALSE.

Value

A Matrix::Matrix() of adjacency status

Examples

```
m <- Matrix::Matrix(stats::runif(1000, 0, 1), 100, 100)
geneset_names <- as.character(stats::runif(100, min = 0, max = 1))
rownames(m) <- colnames(m) <- geneset_names
threshold <- 0.3
adj <- getAdjacencyMatrix(m, threshold)
```

getAnnotation *Get the annotation of a* STRINGdb *object*

Description

Get the annotation of a STRINGdb object, i.e. the aliases of the protein information

Usage

```
getAnnotation(stringdb)
```

Arguments

stringdb	the STRINGdb object

Value

A data.frame mapping STRINGdb ids to gene names

Examples

```
stringdb <- getStringDB(9606)
stringdb
anno_df <- getAnnotation(stringdb)
```

---

getBipartiteGraph *Construct a bipartite graph*

Description

Construct a bipartite graph from cluster information, mapping the cluster to its members

Usage

```
getBipartiteGraph(cluster, gs_names, genes)
```

Arguments

cluster	list, a list of clusters, cluster members are indicated by numeric values.
gs_names	vector, a vector of (geneset) identifiers/names to map the numeric member value in cluster to.
genes	list, a list of vectors of genenames which belong to the genesets in gs_names.

Value

An igraph object to be further manipulated or processed/plotted (e.g. via igraph::plot.igraph() or visNetwork::visIgraph())

Examples

```
cluster <- list(c(1:5), c(6:9))
gs_names <- c("a", "b", "c", "d", "e", "f", "g", "h", "i")
genes <- list(
  c("PDHB", "VARS2"), c("IARS2", "PDHA1"),
  c("AAAS", "ABCE1"), c("ABI1", "AAR2"), c("AATF", "AMFR"),
  c("BMS1", "DAP3"), c("AURKAIP1", "CHCHD1"), c("IARS2"),
  c("AHI1", "ALMS1")
)

g <- getBipartiteGraph(cluster, gs_names, genes)
```

getClusterAdjacencyMatrix *Construct an adjacency matrix*

Description

Construct an adjacency matrix from a `list` of cluster.

Usage

```
getClusterAdjacencyMatrix(cluster, gs_names)
```

Arguments

`cluster`	A `list` of clusters, where each cluster member is indicated by a numeric value
`gs_names`	A vector of geneset names

Value

A `Matrix::Matrix()` of adjacency status

Examples

```
cluster <- list(c(1:5), c(6:9))
gs_names <- c("a", "b", "c", "d", "e", "f", "g", "h", "i")
adj <- getClusterAdjacencyMatrix(cluster, gs_names)
```

getGenes *Split string of genes*

Description

Split a long string of space separated genes into a `list` of individual genes.

Usage

```
getGenes(genesets, gene_name = NULL)
```

Arguments

`genesets`	a `data.frame`, A `data.frame` with at least two columns. One should be called `Geneset`, containing the names/identifiers of the genesets in the data. The second column should be called `Genes` and contains one string of the genes contained in each geneset.
`gene_name`	a character, Alternative name for the column containing the genes in `genesets`. If not given, the column is expected to be called `Genes`.

Value

A `list` containing for each geneset in the `Geneset` column a `list` of the included genes.

`getGraphTitle` *Build up the node title*

Description

Build up the title for the graph nodes to display the available information of each geneset.

Usage

```
getGraphTitle(
  geneset_df = NULL,
  node_ids,
  gs_ids,
  gs_names = NULL,
  cluster_id = NULL
)
```

Arguments

`geneset_df`	A `data.frame` of genesets with a column `Genesets` containing geneset identifiers and a column `Genes` containing the genes belonging to each geneset
`node_ids`	vector, a vector of ids of the nodes in the graph for which the node title should be build.
`gs_ids`	vector, a vector of geneset identifiers, e.g. the `Genesets` column of `geneset_df`.
`gs_names`	vector, a vector of geneset descriptions/names, e.g. the `Term` / `Description` column of `geneset_df`.
`cluster_id`	vector, a vector of cluster ids for each of the genesets

Value

A `list` of titles for a graph with nodes given by `node_ids`.

Examples

```
genes <- list(
  c("PDHB", "VARS2"), c("IARS2", "PDHA1"),
  c("AAAS", "ABCE1"), c("ABI1", "AAR2"), c("AATF", "AMFR"),
  c("BMS1", "DAP3"), c("AURKAIP1", "CHCHD1"), c("IARS2"),
  c("AHI1", "ALMS1")
)
gs_names <- c("a", "b", "c", "d", "e", "f", "g", "h", "i")
geneset_df <- data.frame(
  Genesets = gs_names,
  value = rep(1, 9)
```

```
)
geneset_df$Genes <- genes
graph <- getGraphTitle(
  geneset_df = geneset_df,
  node_ids = c(1:9),
  gs_ids = c(1:9),
  gs_names = gs_names
)
```

---

getId *Get NCBI ID*

Description

Get the NCBI ID of a species

Usage

```
getId(species, version = "12.0", cache = FALSE)
```

Arguments

species	character, the species of your input data
version	character, the version of STRING you want to use, defaults to the current version of STRING
cache	Logical value, defining whether to use the BiocFileCache for retrieval of the files underlying the STRINGdb object. Defaults to TRUE.

Value

A character of the NCBI ID of species

Examples

```
species <- "Homo sapiens"
id <- getId(species = species)

species <- "Mus musculus"
id <- getId(species = species)
```

`getInteractionScore` *Calculate interaction score for two genesets*

Description

The function calculates an interaction score between two sets of genes based on a protein-protein interaction network.

Usage

```
getInteractionScore(a, b, ppi, maxInteract)
```

Arguments

a, b	character vector, set of gene identifiers.
ppi	a `data.frame`, Protein-protein interaction (PPI) network data frame. The object is expected to have three columns, `Gene1` and `Gene2` which specify the gene names of the interacting proteins in no particular order (symmetric interaction) and a column `combined_score` which is a numerical value of the strength of the interaction.
maxInteract	numeric, Maximum interaction value in the PPI.

Value

Interaction score between the two gene sets.

References

See https://doi.org/10.1186/s12864-019-5738-6 for details on the original implementation.

Examples

```
## Mock example showing how the data should look like
a <- c("PDHB", "VARS2", "IARS2")
b <- c("IARS2", "PDHA1")

ppi <- data.frame(
  Gene1 = c("PDHB", "VARS2", "IARS2"),
  Gene2 = c("IARS2", "PDHA1", "CD3"),
  combined_score = c(0.5, 0.2, 0.1)
)
maxInteract <- max(ppi$combined_score)

interaction <- getInteractionScore(a, b, ppi, maxInteract)

## Example using the data available in the package
data(macrophage_topGO_example_small,
     package = "GeDi",
```

```
     envir = environment())
genes <- GeDi::prepareGenesetData(macrophage_topGO_example_small)
data(ppi_macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())
maxInteract <- max(ppi_macrophage_topGO_example_small$combined_score)

interaction <- getInteractionScore(genes[1], genes[2], ppi, maxInteract)
```

`getJaccardMatrix` *Get Matrix of Jaccard distances*

Description

Calculate the Jaccard distance of all combinations of genesets in a given data set of genesets.

Usage

```
getJaccardMatrix(genesets)
```

Arguments

`genesets`	a `list`, A `list` of genesets where each genesets is represented by `list` of genes.

Value

A `Matrix::Matrix()` with Jaccard distance rounded to 2 decimal places.

Examples

```
## Mock example showing how the data should look like
genesets <- list(list("PDHB", "VARS2"), list("IARS2", "PDHA1"))
m <- getJaccardMatrix(genesets)

## Example using the data available in the package
data(macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())
genes <- GeDi::prepareGenesetData(macrophage_topGO_example_small)
jaccard <- getJaccardMatrix(genes)
```

getKappaMatrix *Get Matrix of Kappa distances*

Description

Calculate the Kappa distance of all combinations of genesets in a given data set of genesets. The Kappa distance is normalized to the (0, 1) interval.

Usage

```
getKappaMatrix(genesets)
```

Arguments

genesets	a list, A list of genesets where each genesets is represented by list of genes.

Value

A Matrix::Matrix() with Kappa distance rounded to 2 decimal places.

Examples

```
#' ## Mock example showing how the data should look like
genesets <- list(list("PDHB", "VARS2"), list("IARS2", "PDHA1"))
m <- getKappaMatrix(genesets)

## Example using the data available in the package
data(macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())
genes <- GeDi::prepareGenesetData(macrophage_topGO_example_small)
kappa <- getKappaMatrix(genes)
```

getMeetMinMatrix *Get Matrix of Meet-Min distances*

Description

Calculate the Meet-Min distance of all combinations of genesets in a given data set of genesets.

Usage

```
getMeetMinMatrix(genesets)
```

Arguments

genesets	a list, A list of genesets where each genesets is represented by list of genes.

Value

A Matrix::Matrix() with Meet-Min distance rounded to 2 decimal places.

Examples

```
## Mock example showing how the data should look like
genesets <- list(list("PDHB", "VARS2"), list("IARS2", "PDHA1"))
m <- getMeetMinMatrix(genesets)

## Example using the data available in the package
data(macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())
genes <- GeDi::prepareGenesetData(macrophage_topGO_example_small)
mm <- getMeetMinMatrix(genes)
```

---

getpMMMatrix *Calculate the pMM distance*

Description

Calculate the pMM distance of all combinations of genesets in a given data set of genesets.

Usage

```
getpMMMatrix(
  genesets,
  ppi,
  alpha = 1,
  progress = NULL,
  BPPARAM = BiocParallel::SerialParam()
)
```

Arguments

genesets	a list, A list of genesets where each genesets is represented by list of genes.
ppi	a data.frame, Protein-protein interaction (PPI) network data frame. The object is expected to have three columns, Gene1 and Gene2 which specify the gene names of the interacting proteins in no particular order (symmetric interaction) and a column combined_score which is a numerical value of the strength of the interaction.
alpha	numeric, Scaling factor for controlling the influence of the interaction score. Defaults to 1.
progress	a shiny::Progress() object, Optional progress bar object to track the progress of the function (e.g. in a Shiny app).
BPPARAM	A BiocParallel bpparam object specifying how parallelization should be handled. Defaults to BiocParallel::SerialParam()

Value

A Matrix::Matrix() with pMM distance rounded to 2 decimal places.

References

See https://doi.org/10.1186/s12864-019-5738-6 for details on the original implementation.

Examples

```
## Mock example showing how the data should look like
genesets <- list(c("PDHB", "VARS2"), c("IARS2", "PDHA1"))

ppi <- data.frame(
  Gene1 = c("PDHB", "VARS2"),
  Gene2 = c("IARS2", "PDHA1"),
  combined_score = c(0.5, 0.2)
)

pMM <- getpMMMatrix(genesets, ppi)

## Example using the data available in the package
data(macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())
genes <- GeDi::prepareGenesetData(macrophage_topGO_example_small)
data(ppi_macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())

pMM <- getpMMMatrix(genes, ppi)
```

---

getPPI *Download Protein-Protein Interaction (PPI)*

Description

Download the Protein-Protein Interaction (PPI) information of a STRINGdb object

Usage

```
getPPI(genes, stringdb, anno_df)
```

Arguments

genes	a list, A list of genes to download the respective protein- protein interaction information
stringdb	A STRINGdb object, the species of the object should match the species of genes.
anno_df	An annotation data.frame mapping STRINGdb ids to gene names, e.g. downloaded with GeDi::getAnnotation()

Value

A `data.frame` of Protein-Protein interactions

Examples

```
## Mock example showing how the data should look like

genes <- c(c("CFTR", "RALA"), c("CACNG3", "ITGA3"), c("DVL2"))
stringdb <- getStringDB(9606, cache_location = FALSE)
# stringdb
anno_df <- getAnnotation(stringdb)
ppi <- getPPI(genes, stringdb, anno_df)

## Example using the data available in the package
## Not run:
data(macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())
stringdb <- getStringDB(9606)
stringdb
anno_df <- getAnnotation(stringdb)
genes <- GeDi::prepareGenesetData(macrophage_topGO_example_small)
ppi <- getPPI(genes, stringdb, anno_df)

## End(Not run)
```

`getSorensenDiceMatrix` *Get Matrix of Sorensen-Dice distances*

Description

Calculate the Sorensen-Dice distance of all combinations of genesets in a given data set of genesets.

Usage

```
getSorensenDiceMatrix(genesets)
```

Arguments

`genesets`	a `list`, A `list` of genesets where each geneset is represented by `list` of genes.

Value

A `Matrix::Matrix()` with Sorensen-Dice distance rounded to 2 decimal places.

Examples

```
## Mock example showing how the data should look like
genesets <- list(list("PDHB", "VARS2"), list("IARS2", "PDHA1"))
m <- getSorensenDiceMatrix(genesets)

## Example using the data available in the package
data(macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())
genes <- GeDi::prepareGenesetData(macrophage_topGO_example_small)
sd_matrix <- getSorensenDiceMatrix(genes)
```

`getStringDB` *Get the STRING db entry of a species*

Description

Get the respective STRINGdb object of your species of interest

Usage

```
getStringDB(
  species,
  version = "12.0",
  score_threshold = 0,
  cache_location = FALSE
)
```

Arguments

species	numeric, the NCBI ID of the species of interest
version	character, The STRINGdb version to use, defaults to the current version
score_threshold	numeric, A score threshold to cut the retrieved interactions, defaults to 0 (all interactions)
cache_location	Logical value, defining whether to use the BiocFileCache for retrieval of the files underlying the STRINGdb object. Defaults to TRUE.

Value

a STRINGdb object of species

Examples

```
species <- getId(species = "Homo sapiens")
stringdb <- getStringDB(as.numeric(species))
```

goDistance *Calculate similarity of GO terms*

Description

Calculate the pairwise similarity of GO terms

Usage

```
goDistance(
  geneset_ids,
  method = "Wang",
  ontology = "BP",
  species = "org.Hs.eg.db"
)
```

Arguments

geneset_ids	list, a list of GO identifiers to score
method	character, the method to calculate the GO distance. Possible options are "Resnik", "Lin", "Rel", "Jiang", "Wang".
ontology	character, the ontology to use. Possible options are "BP", "MF" and "CC".
species	character, the species of your data. Indicated as org.XX.eg.db package from Bioconductor.

Value

A Matrix::Matrix() with the pairwise GO distance of each geneset pair.

Examples

```
## Mock example showing how the data should look like
go_ids <- c("GO:0002503", "GO:0045087", "GO:0019886",
            "GO:0002250", "GO:0001916", "GO:0019885")

similarity <- goDistance(go_ids)

## Example using the data available in the package
data(macrophage_topGO_example_small, package = "GeDi")
go_ids <- macrophage_topGO_example_small$Genesets
## Not run:
similarity <- goDistance(go_ids)

## End(Not run)
```

gsHistogram *Create a histogram plot for gene set sizes*

Description

Create a histogram plot to plot geneset names / identifiers against their size.

Usage

```
gsHistogram(
  genesets,
  gs_names,
  gs_description = NULL,
  start = 0,
  end = 0,
  binwidth = 5,
  color = "#0092AC"
)
```

Arguments

genesets	a list, A list of genesets where each genesets is represented by list of genes.
gs_names	character vector, Name / identifier of the genesets in genesets
gs_description	Optional, a character vector containing a short description for each geneset
start	numeric, Optional, describes the minimum gene set size to include. Defaults to 0.
end	numeric, Optional, describes the maximum gene set size to include. Defaults to 0.
binwidth	numeric, Width of histogram bins. Defaults to 5.
color	character, Fill color for histogram bars. Defaults to #0092AC.

Value

A ggplot2::ggplot() plot object.

Examples

```
## Mock example showing how the data should look like
gs_names <- c("a", "b", "c", "d", "e", "f", "g", "h")
genesets <- list(
  c("PDHB", "VARS2", "IARS2", "PDHA1"),
  c("AAAS", "ABCE1"), c("ABI1", "AAR2", "AATF"), c("AMFR"),
  c("BMS1", "DAP3"), c("AURKAIP1", "CHCHD1"), c("IARS2"),
  c("AHI1", "ALMS1")
)
```

```
p <- gsHistogram(genesets, gs_names, binwidth = 1)

## Example using the data available in the package
data(macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())
genes <- GeDi::prepareGenesetData(macrophage_topGO_example_small)
p <- gsHistogram(genes, macrophage_topGO_example_small$Genesets)
```

`kMeansClustering` *Calculate clusters based on kMeans clustering*

Description

This function performs kMeans clustering on a set of scores.

Usage

```
kMeansClustering(scores, k, iter = 500, nstart = 50)
```

Arguments

`scores`	A `Matrix::Matrix()` of (distance) scores
`k`	numerical, the number of centers to start with. This number will correlate with the resulting number of clusters.
`iter`	numerical, number of iterations for refinement. Defaults to 500.
`nstart`	numerical, how often the start points should be switched. Ensures a robust clustering, as clustering is influenced by the start points. Defaults to 50.

Value

A `list` of clusters

Examples

```
## Mock example showing how the data should look like
scores <- Matrix::Matrix(stats::runif(100, min = 0, max = 1), 10, 10)
rownames(scores) <- colnames(scores) <- c("a", "b", "c", "d", "e",
                                 "f", "g", "h", "i", "j")
cluster <- kMeansClustering(scores, k = 3)

## Example using the data available in the package
data(scores_macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())

cluster <- kMeansClustering(scores_macrophage_topGO_example_small,
                            k = 5)
```

kNN_clustering *Calculate clusters based on kNN clustering*

Description

This function performs k-Nearest Neighbors (kNN) clustering on a set of scores.

Usage

```
kNN_clustering(scores, k)
```

Arguments

scores	A Matrix::Matrix() of (distance) scores
k	numerical, the number of neighbors

Value

A list of clusters

Examples

```
## Mock example showing how the data should look like
scores <- Matrix::Matrix(stats::runif(100, min = 0, max = 1), 10, 10)
rownames(scores) <- colnames(scores) <- c("a", "b", "c", "d", "e",
                                  "f", "g", "h", "i", "j")
cluster <- kNN_clustering(scores, k = 3)

## Example using the data available in the package
data(scores_macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())

kNN <- kNN_clustering(scores_macrophage_topGO_example_small,
                        k = 5)
```

louvainClustering *Cluster genesets using Louvain clustering.*

Description

This function is a wrapper function for the Louvain clustering. The actual computation of the clustering is done in the GeDi::clustering() function. This function is mainly a wrapper function for stand-alone use of GeDi functionalities to enhance user experience and allow for a clearer distinction of the individual clustering algorithms.

Usage

```
louvainClustering(scores, threshold)
```

Arguments

scores	A Matrix::Matrix() of (distance) scores
threshold	numerical, A threshold used to determine which genesets are considered similar. Genesets are considered similar if (distance) score <= threshold. similar.

Value

A list of clusters

Examples

```
## Mock example showing how the data should look like
m <- Matrix::Matrix(stats::runif(100, min = 0, max = 1), 10, 10)
rownames(m) <- colnames(m) <- c("a", "b", "c", "d", "e",
                                "f", "g", "h", "i", "j")
louvainCluster <- louvainClustering(m, 0.3)

## Example using the data available in the package
data(scores_macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())

louvainCluster <- louvainClustering(scores_macrophage_topGO_example_small,
                        threshold = 0.5)
```

---

macrophage_KEGG_example

A sample input RData file

Description

A sample input RData file generated from the macrophage dataset.

Format

A data.frame object

Details

This sample input contains data from the macrophage package found on Bioconductor. The exact steps used to generated this file can be found in the package vignette. The used database for the enrichment was the KEGG database.

References

Alasoo, K., Rodrigues, J., Mukhopadhyay, S. et al. Shared genetic effects on chromatin and gene expression indicate a role for enhancer priming in immune response. Nat Genet 50, 424–431 (2018). https://doi.org/10.1038/s41588-018-0046-7

`macrophage_Reactome_example`

A sample input RData file

Description

A sample input RData file generated from the macrophage dataset.

Format

A data.frame object

Details

This sample input contains data from the macrophage package found on Bioconductor. The exact steps used to generated this file can be found in the package vignette. The used database for the enrichment was the Reactome database.

References

Alasoo, K., Rodrigues, J., Mukhopadhyay, S. et al. Shared genetic effects on chromatin and gene expression indicate a role for enhancer priming in immune response. Nat Genet 50, 424–431 (2018). https://doi.org/10.1038/s41588-018-0046-7

`macrophage_topGO_example`

A sample input RData file

Description

A sample input RData file generated from the macrophage dataset.

Format

A data.frame object

Details

This sample input contains data from the macrophage package found on Bioconductor. The exact steps used to generated this file can be found in the package vignette.

References

Alasoo, K., Rodrigues, J., Mukhopadhyay, S. et al. Shared genetic effects on chromatin and gene expression indicate a role for enhancer priming in immune response. Nat Genet 50, 424–431 (2018). https://doi.org/10.1038/s41588-018-0046-7

---

macrophage_topGO_example_small

A small sample input RData file

Description

A small sample input RData file generated from the macrophage dataset.

Format

A data.frame object

Details

This sample input contains data from the macrophage package found on Bioconductor. It is a small version of the `macrophage_topGO_example` and only contains the first 50 rows of this example. It can be used for fast testing of the application.

References

Alasoo, K., Rodrigues, J., Mukhopadhyay, S. et al. Shared genetic effects on chromatin and gene expression indicate a role for enhancer priming in immune response. Nat Genet 50, 424–431 (2018). https://doi.org/10.1038/s41588-018-0046-7

---

markovClustering *Cluster genesets using Markov clustering.*

Description

This function is a wrapper function for the Markov clustering. The actual computation of the clustering is done in the `GeDi::clustering()` function. This function is mainly a wrapper function for stand-alone use of GeDi functionalities to enhance user experience and allow for a clearer distinction of the individual clustering algorithms.

Usage

```
markovClustering(scores, threshold)
```

Arguments

scores	A Matrix::Matrix() of (distance) scores
threshold	numerical, A threshold used to determine which genesets are considered similar. Genesets are considered similar if (distance) score <= threshold. similar.

Value

A list of clusters

Examples

```
## Mock example showing how the data should look like
m <- Matrix::Matrix(stats::runif(100, min = 0, max = 1), 10, 10)
rownames(m) <- colnames(m) <- c("a", "b", "c", "d", "e",
                                "f", "g", "h", "i", "j")
markovCluster <- markovClustering(m, 0.3)

## Example using the data available in the package
data(scores_macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())

markovCluster <- markovClustering(scores_macrophage_topGO_example_small,
                        threshold = 0.5)
```

---

pamClustering *Calculate clusters based on PAM clustering*

Description

This function performs Partioning aroung Medoids clustering on a set of scores.

Usage

```
pamClustering(scores, k)
```

Arguments

scores	A Matrix::Matrix() of (distance) scores
k	numerical, the number of centers to start with. This number will correlate with the resulting number of clusters.

Value

A list of clusters

Examples

```
## Mock example showing how the data should look like
scores <- Matrix::Matrix(stats::runif(100, min = 0, max = 1), 10, 10)
rownames(scores) <- colnames(scores) <- c("a", "b", "c", "d", "e",
                                "f", "g", "h", "i", "j")
cluster <- pamClustering(scores, k = 3)

## Example using the data available in the package
data(scores_macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())

cluster <- pamClustering(scores_macrophage_topGO_example_small,
                                k = 5)
```

`path_to_GeDi` *Prepare enrichment analysis output data for GeDi*

Description

Prepare enrichment analysis output data for GeDi

Usage

```
path_to_GeDi(genesets_df, enrichment_package)
```

Arguments

`genesets_df`	Dataframe, a dataframe of the enrichment analysis output which should be prepared for the use in GeDi.
`enrichment_package`	character, the name of the R package used to conduct the enrichment analysis. Supported packages are currently topGO, clusterProfiler, ReactomePA, enrichR and fgsea.

Value

A Dataframe of the inout data which can directly be used with the GeDi package.

Examples

```
genesets_df <- data("macrophage_Reactome_example", package = "GeDi")
genesets_df <- path_to_GeDi(
               genesets_df = macrophage_Reactome_example,
               enrichment_package = "clusterProfiler")
```

pMMlocal *Calculate local pMM distance*

Description

Calculate the local pMM distance of two genesets.

Usage

```
pMMlocal(a, b, ppi, maxInteract, alpha)
```

Arguments

a, b	character vector, set of gene identifiers.
ppi	a data.frame, Protein-protein interaction (PPI) network data frame. The object is expected to have three columns, Gene1 and Gene2 which specify the gene names of the interacting proteins in no particular order (symmetric interaction) and a column combined_score which is a numerical value of the strength of the interaction.
maxInteract	numeric, Maximum interaction value in the PPI.
alpha	numeric, Scaling factor for controlling the influence of the interaction score. Defaults to 1.

Value

The pMMlocal score between the two gene sets.

References

See https://doi.org/10.1186/s12864-019-5738-6 for details on the original implementation.

Examples

```
## Mock example showing how the data should look like
a <- c("PDHB", "VARS2")
b <- c("IARS2", "PDHA1")

ppi <- data.frame(
  Gene1 = c("PDHB", "VARS2"),
  Gene2 = c("IARS2", "PDHA1"),
  combined_score = c(0.5, 0.2)
)
maxInteract <- max(ppi$combined_score)

pMM_score <- pMMlocal(a, b, ppi, alpha = 1,  maxInteract)

## Example using the data available in the package
data(macrophage_topGO_example_small,
```

```
     package = "GeDi",
     envir = environment())
genes <- GeDi::prepareGenesetData(macrophage_topGO_example_small)
data(ppi_macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())
maxInteract <- max(ppi_macrophage_topGO_example_small$combined_score)

pMMlocal <- pMMlocal(genes[1], genes[2], ppi, alpha = 1,  maxInteract)
```

`ppi_macrophage_topGO_example_small` *PPI*

Description

A file containing a Protein-Protein Interaction (PPI) `data.frame` for the `macrophage_topGO_example_small`.

Format

A data.frame object

Details

This sample input contains a PPI for the `macrophage_topGO_example_small`. The PPI has been downloaded using the functions to download a PPI matrix. Please check out the vignette for further information.

References

Alasoo, K., Rodrigues, J., Mukhopadhyay, S. et al. Shared genetic effects on chromatin and gene expression indicate a role for enhancer priming in immune response. Nat Genet 50, 424–431 (2018). https://doi.org/10.1038/s41588-018-0046-7

`prepareGenesetData` *Split string of genes*

Description

Split a long string of space separated genes into a `list` of individual genes.

Usage

```
prepareGenesetData(genesets, gene_name = NULL)
```

Arguments

genesets	a data.frame, A data.frame with at least two columns. One should be called Geneset, containing the names/identifiers of the genesets in the data. The second column should be called Genes and contains one string of the genes contained in each geneset.
gene_name	a character, Alternative name for the column containing the genes in genesets. If not given, the column is expected to be called Genes.

Value

A list containing for each geneset in the Geneset column a list of the included genes.

Examples

```
## Mock example showing how the data should look like
df <- data.frame(
  Geneset = c(
    "Cell Cycle",
    "Biological Process",
    "Mitosis"
  ),
  Genes = c(
    c("PDHB,VARS2,IARS2"),
    c("LARS,LARS2"),
    c("IARS,SUV3")
  )
)
genes <- prepareGenesetData(df)

## Example using the data available in the package
data(macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())
genes <- prepareGenesetData(macrophage_topGO_example_small)
```

---

sample_geneset *A sample input text file*

Description

A sample input text file taken from the GScluster package

Format

Text file

Details

This sample input text file contains data from the GScluster package. It is identical to the sample_geneset.txt file found on the Github page of the package.

References

Yoon, S., Kim, J., Kim, SK. et al. GScluster: network-weighted gene-set clustering analysis. BMC Genomics 20, 352 (2019). https://doi.org/10.1186/s12864-019-5738-6

---

sample_geneset_broken *A broken input text file*

Description

A broken input text file to test the application

Format

Text file

Details

This sample input text file is broken and used for testing the application.

---

sample_geneset_empty *An empty input text file*

Description

An empty input text file to test the application

Format

Text file

Details

This sample input text file is empty and used for testing the application.

sample_geneset_small *A small sample input text file*

Description

A sample input text file taken from the GScluster package, which is reduced to a smaller number of entries for faster testing of the application.

Format

Text file

Details

This sample input text file contains data from the GScluster package. It was taken from the sample_geneset.txt file found on the Github page of the package and then reduced to a smaller amount of entries for faster testing of the application.

References

Yoon, S., Kim, J., Kim, SK. et al. GScluster: network-weighted gene-set clustering analysis. BMC Genomics 20, 352 (2019). https://doi.org/10.1186/s12864-019-5738-6

scores_macrophage_topGO_example_small *Sample scores*

Description

A file containing sample distance scores for the `macrophage_topGO_example_small`.

Format

A sparse matrix (`dgCMatrix`)

Details

This sample input contains scores for the `macrophage_topGO_example_small`. Distance scores have been calculated using the `getJaccardMatrix()` method.

References

Alasoo, K., Rodrigues, J., Mukhopadhyay, S. et al. Shared genetic effects on chromatin and gene expression indicate a role for enhancer priming in immune response. Nat Genet 50, 424–431 (2018). https://doi.org/10.1038/s41588-018-0046-7

`seedFinding` *Find clustering seeds*

Description

Determine initial seeds for the clustering from the distance score matrix.

Usage

```
seedFinding(distances, simThreshold, memThreshold)
```

Arguments

`distances`	A `Matrix::Matrix()` of (distance) scores
`simThreshold`	numerical, A threshold to determine which genesets are considered close (i.e. have a distance <= simThreshold) in the `distances` matrix.
`memThreshold`	numerical, A threshold used to ensure that enough members of a potential seed set are close/similar to each other. Only if this condition is met, the set is considered a seed.

Value

A `list` of seeds which can be used for clustering

References

See https://david.ncifcrf.gov/helps/functional_classification.html#clustering for details on the original implementation

Examples

```
## Mock example showing how the data should look like

m <- Matrix::Matrix(stats::runif(100, min = 0, max = 1), 10, 10)
seeds <- seedFinding(distances = m, simThreshold = 0.3, memThreshold = 0.5)

## Example using the data available in the package
data(scores_macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())

seeds <- seedFinding(scores_macrophage_topGO_example_small,
                     simThreshold = 0.3,
                     memThreshold = 0.5)
```

Index

.checkGTL, 4
.checkGenesets, 3
.checkPPI, 4
.checkScores, 5
.cluster_markov, 5
.filterGenesets, 7
.findSeparator, 7
.getClusterDatatable, 8
.getGenesetDescriptions, 8
.getNumberCores, 9
.graphMetricsGenesetsDT, 9
.map_to_color, 10
.sepguesser, 11

BiocParallel::SerialParam(), *31*
buildClusterGraph, 11
buildGraph, 12
buildHistogramData, 13

checkInclusion, 14
clustering, 15
ComplexHeatmap::Heatmap(), *18*

data.frame, *4*
deprecated, 16
distanceDendro, 17
distanceHeatmap, 17

enrichmentWordcloud, 19

fuzzyClustering, 20

GeDi, 21, *43*
GeneTonic::GeneTonic_list(), *4*, *22*
getAdjacencyMatrix, 23
getAnnotation, 23
getBipartiteGraph, 24
getClusterAdjacencyMatrix, 25
getGenes, 25
getGenes(), *16*
getGraphTitle, 26
getId, 27
getInteractionScore, 28
getJaccardMatrix, 29
getJaccardMatrix(), *48*
getKappaMatrix, 30
getMeetMinMatrix, 30
getpMMMatrix, 31
getPPI, 32
getSorensenDiceMatrix, 33
getStringDB, 34
ggdendro::ggdendrogram(), *17*
ggplot2::ggplot(), *36*
goDistance, 35
gsHistogram, 36

igraph::communities(), *6*
igraph::plot.igraph(), *12, 13*, *24*

kMeansClustering, 37
kNN_clustering, 38

louvainClustering, 38

macrophage_KEGG_example, 39
macrophage_Reactome_example, 40
macrophage_topGO_example, 40
macrophage_topGO_example_small, 41
markovClustering, 41
Matrix::Matrix(), *5*, *13*, *15*, *17, 18*, *22, 23*, *25*, *29–33*, *35*, *37–39*, *42*, *49*

pamClustering, 42
path_to_GeDi, 43
pMMlocal, 44
ppi_macrophage_topGO_example_small, 45
prepareGenesetData, 45
prepareGenesetData(), *16*

RColorBrewer::brewer.pal(), *10*

sample_geneset, 46

sample_geneset_broken, 47
sample_geneset_empty, 47
sample_geneset_small, 48
scores_macrophage_topGO_example_small, 48
seedFinding, 49
shiny::Progress(), *31*
stats::hclust(), *17*

visNetwork::visIgraph(), *12, 13*, *24*

wordcloud2::wordcloud2(), *19*