

# Package ‘GeDi’

December 23, 2024

**Title** Defining and visualizing the distances between different genesets

**Version** 1.2.0

**Date** 2024-10-10

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

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0), DESeq2, htmltools, pcaExplorer, 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** 7.3.2

**Roxygen** list(markdown = TRUE)

**Config/testthat/edition** 3

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

**git\_url** <https://git.bioconductor.org/packages/GeDi>

**git\_branch** RELEASE\_3\_20

**git\_last\_commit** abf007a

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.20

**Date/Publication** 2024-12-23

**Author** Annekathrin Nedwed [aut, cre] (<<https://orcid.org/0000-0002-2475-4945>>),  
 Federico Marini [aut] (<<https://orcid.org/0000-0003-3252-7758>>)

**Maintainer** Annekathrin Nedwed <anneludt@uni-mainz.de>

## Contents

.checkGenesets	3
.checkGTL	4
.checkPPI	4
.checkScores	5
.filterGenesets	5
.findSeparator	6
.getClusterDatatable	6
.getGenesetDescriptions	7
.getNumberCores	7
.graphMetricsGenesetsDT	8
.sepguesser	8
buildClusterGraph	9
buildGraph	10
buildHistogramData	11
calculateJaccard	12
calculateKappa	12
calculateSorensenDice	13
checkInclusion	14
clustering	14
deprecated	15
distanceDendro	16
distanceHeatmap	17
enrichmentWordcloud	18
fuzzyClustering	19
GeDi	20
getAdjacencyMatrix	21
getAnnotation	22
getBipartiteGraph	22
getClusterAdjacencyMatrix	23
getGenes	23
getGraphTitle	24
getId	25
getInteractionScore	26
getJaccardMatrix	27
getKappaMatrix	28
getMeetMinMatrix	28
getpMMMatrix	29
getPPI	31
getSorensenDiceMatrix	32
getStringDB	32
goDistance	33
gsHistogram	34
kMeansClustering	35

kNN\_clustering . . . . . 36

louvainClustering . . . . . 37

macrophage\_KEGG\_example . . . . . 38

macrophage\_Reactome\_example . . . . . 38

macrophage\_topGO\_example . . . . . 39

macrophage\_topGO\_example\_small . . . . . 39

markovClustering . . . . . 40

pamClustering . . . . . 40

pMMlocal . . . . . 41

ppi\_macrophage\_topGO\_example\_small . . . . . 42

prepareGenesetData . . . . . 43

sample\_geneset . . . . . 44

sample\_geneset\_broken . . . . . 44

sample\_geneset\_empty . . . . . 44

sample\_geneset\_small . . . . . 45

scaleGO . . . . . 45

scores\_macrophage\_topGO\_example\_small . . . . . 46

seedFinding . . . . . 47

**Index** **48**

---

.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(gt1)
```

### Arguments

gt1 A GeneTonicList object 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()`.

---

.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

<code>g</code>	A <a href="#">igraph</a> graph object
<code>genesets</code>	A data.frame of genesets with a column <code>Genesets</code> containing geneset identifiers and a column <code>Genes</code> 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.

---

```
.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

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

### Value

A 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.



---

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 <code>Matrix::Matrix()</code> 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

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

---

calculateJaccard      *Calculate the Jaccard distance*

---

**Description**

Calculate the Jaccard distance between two genesets.

**Usage**

```
calculateJaccard(a, b)
```

**Arguments**

a, b                      character vector, set of gene identifiers.

**Value**

The Jaccard distance of the sets.

**Examples**

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

## 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 <- calculateJaccard(genes[1], genes[2])
```

---

calculateKappa      *Calculate the Kappa distance*

---

**Description**

Calculate the Kappa distance between two genesets.

**Usage**

```
calculateKappa(a, b, all_genes)
```

**Arguments**

a, b                      character vector, set of gene identifiers.  
all\_genes                character vector, list of all (unique) genes available in the input data.

**Value**

The Kappa distance of the sets.

## Examples

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

## Example using the data available in the package
data(macrophage_topGO_example_small,
     package = "GeDi",
     envir = environment())
genes <- GeDi::prepareGenesetData(macrophage_topGO_example_small)
c <- calculateKappa(genes[1], genes[2], unique(genes))
```

---

calculateSorensenDice *Calculate the Sorensen-Dice distance*

---

## Description

Calculate the Sorensen-Dice distance between two genesets.

## Usage

```
calculateSorensenDice(a, b)
```

## Arguments

a, b                    character vector, set of gene identifiers.

## Value

The Sorensen-Dice distance of the sets.

## Examples

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

## 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 <- calculateSorensenDice(genes[1], genes[2])
```

---

checkInclusion	<i>Check for subset inclusion</i>
----------------	-----------------------------------

---

**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	<i>Cluster genesets.</i>
------------	--------------------------

---

**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  $\leq$  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 function 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	<i>Plot a dendrogram</i>
----------------	--------------------------

---

**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"
)
```

### Arguments

distance_scores	A <code>Matrix::Matrix()</code> 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 column labels 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 columns should be clustered based on the distance scores. Defaults to TRUE
title	character, a title for the figure. Defaults to "Distance Scores"

### 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)
```

## 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.

## 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 <code>GeDi::seedFinding()</code> function
threshold	numerical, A threshold for merging seeds

## Value

A list of clusters

## References

See [https://david.ncicrf.gov/helps/functional\\_classification.html#clustering](https://david.ncicrf.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 <code>data.frame</code> , The input data used for GeDi. This should be a <code>data.frame</code> 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 <code>data.frame</code> , 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 <code>combined_score</code> which is a numerical value of the strength of the interaction.
distance_scores	A <code>Matrix::Matrix()</code> of (distance) scores
gtl	A <code>GeneTonicList</code> object generated with <code>GeneTonic::GeneTonic_list()</code> , 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	<i>Get the annotation of a <a href="#">STRINGdb</a> object</i>
---------------	--

---

**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	<i>Construct a bipartite graph</i>
-------------------	------------------------------------

---

**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	<i>Build up the node title</i>
---------------	--------------------------------

---

**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 <a href="#">STRINGdb</a> 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,
  progress = NULL,
  BPPARAM = BiocParallel::SerialParam()
)
```

## Arguments

genesets	a list, A list of genesets where each genesets is represented by list of genes.
progress	a <a href="#">shiny::Progress()</a> 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 <a href="#">BiocParallel::SerialParam()</a>

## 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,
  progress = NULL,
  BPPARAM = BiocParallel::SerialParam()
)
```

### Arguments

genesets	a list, A list of genesets where each genesets is represented by list of genes.
progress	a <a href="#">shiny::Progress()</a> 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 <a href="#">BiocParallel::SerialParam()</a>

### 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,
  progress = NULL,
  BPPARAM = BiocParallel::SerialParam()
)
```

**Arguments**

genesets	a list, A list of genesets where each genesets is represented by list of genes.
progress	a <code>shiny::Progress()</code> object, Optional progress bar object to track the progress of the function (e.g. in a Shiny app).
BPPARAM	A <code>BiocParallel</code> bpparam object specifying how parallelization should be handled. Defaults to <code>BiocParallel::SerialParam()</code>

**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)
```

---

getpMMMMatrix

*Calculate the pMM distance*


---

**Description**

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

**Usage**

```
getpMMMMatrix(
  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 <code>shiny::Progress()</code> 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 <code>BiocParallel::SerialParam()</code>

**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 <- getpMMMMatrix(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 <- getpMMMMatrix(genes, ppi)
```

---

getPPI	<i>Download Protein-Protein Interaction (PPI)</i>
--------	---

---

## 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 <a href="#">STRINGdb</a> object, the species of the object should match the species of genes.
anno_df	An annotation data.frame mapping <a href="#">STRINGdb</a> ids to gene names, e.g. downloaded with <code>GeDi::getAnnotation()</code>

## 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,  
  progress = NULL,  
  BPPARAM = BiocParallel::SerialParam()  
)
```

### Arguments

genesets	a list, A list of genesets where each genesets is represented by list of genes.
progress	a <a href="#">shiny::Progress()</a> object, Optional progress bar object to track the progress of the function (e.g. in a Shiny app).
BPPARAM	A <a href="#">BiocParallel</a> bpparam object specifying how parallelization should be handled. Defaults to <a href="#">BiocParallel::SerialParam()</a>

### 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",  
  progress = NULL,  
  BPPARAM = BiocParallel::SerialParam()  
)
```

**Arguments**

geneset_ids	list, a list of GO identifiers to score
method	character, the method to calculate the GO distance. See <a href="#">GOSemSim::goSim</a> measure parameter for possibilities.
ontology	character, the ontology to use. See <a href="#">GOSemSim::goSim</a> ont parameter for possibilities.
species	character, the species of your data. Indicated as org.XX.eg.db package from Bioconductor.
progress	<a href="#">shiny::Progress()</a> object, optional. 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 <a href="#">BiocParallel::SerialParam()</a>

**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 <code>Matrix::Matrix()</code> 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) <- 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 <code>Matrix::Matrix()</code> 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 <code>Matrix::Matrix()</code> of (distance) scores
threshold	numerical, A threshold used to determine which genesets are considered similar. Genesets are considered similar if (distance) score $\leq$ 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 generate 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 generate 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 generate 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	<i>Cluster genesets using Markov clustering.</i>
------------------	--

---

### 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 <code>Matrix::Matrix()</code> of (distance) scores
threshold	numerical, A threshold used to determine which genesets are considered similar. Genesets are considered similar if (distance) score $\leq$ 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	<i>Calculate clusters based on PAM clustering</i>
---------------	---

---

### Description

This function performs Partitioning around 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)
```

---

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>

---

scaleGO *Scaling (distance) scores*

---

### Description

A method to scale a matrix of distance scores with the GO term similarity of the associated genesets.

### Usage

```
scaleGO(
  scores,
  geneset_ids,
  method = "Wang",
  ontology = "BP",
  species = "org.Hs.eg.db",
  BPPARAM = BiocParallel::SerialParam()
)
```

### Arguments

scores	a <code>Matrix::Matrix()</code> , a matrix of (distance) scores for the identifiers in geneset_ids.
geneset_ids	list, a list of GO identifiers to score
method	character, the method to calculate the GO distance. See <code>GOSemSim::goSim</code> measure parameter for possibilities.
ontology	character, the ontology to use. See <code>GOSemSim::goSim</code> ont parameter for possibilities.

species	character, the species of your data. Indicated as org.XX.eg.db package from Bioconductor.
BPPARAM	A BiocParallelParam object specifying how parallelization should be handled

### Value

A `Matrix::Matrix()` of scaled values.

### 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")
set.seed(42)
scores <- Matrix::Matrix(stats::runif(36, min = 0, max = 1), 6, 6)
similarity <- scaleGO(scores,
                      go_ids)

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

## End(Not run)
```

---

```
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	<i>Find clustering seeds</i>
-------------	------------------------------

---

### Description

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

### Usage

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

### Arguments

distances	A <code>Matrix::Matrix()</code> of (distance) scores
simThreshold	numerical, A threshold to determine which genesets are considered close (i.e. have a distance $\leq$ 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](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  
.filterGenesets, 5  
.findSeparator, 6  
.getClusterDatatable, 6  
.getGenesetDescriptions, 7  
.getNumberCores, 7  
.graphMetricsGenesetsDT, 8  
.sepguesser, 8  
BiocParallel::SerialParam(), 27–30, 32, 34  
buildClusterGraph, 9  
buildGraph, 10  
buildHistogramData, 11  
calculateJaccard, 12  
calculateKappa, 12  
calculateSorensenDice, 13  
checkInclusion, 14  
clustering, 14  
ComplexHeatmap::Heatmap(), 17  
data.frame, 4  
deprecated, 15  
distanceDendro, 16  
distanceHeatmap, 17  
enrichmentWordcloud, 18  
fuzzyClustering, 19  
GeDi, 20  
GeneTonic::GeneTonic\_list(), 4, 20  
getAdjacencyMatrix, 21  
getAnnotation, 22  
getBipartiteGraph, 22  
getClusterAdjacencyMatrix, 23  
getGenes, 23  
getGenes(), 15  
getGraphTitle, 24  
getId, 25  
getInteractionScore, 26  
getJaccardMatrix, 27  
getJaccardMatrix(), 46  
getKappaMatrix, 28  
getMeetMinMatrix, 28  
getpMMMatrix, 29  
getPPI, 31  
getSorensenDiceMatrix, 32  
getStringDB, 32  
ggdendro::ggdendrogram(), 16  
ggplot2::ggplot(), 35  
goDistance, 33  
GOSemSim::goSim, 34, 45  
gsHistogram, 34  
igraph, 8, 9  
igraph::plot.igraph(), 9, 10, 22  
kMeansClustering, 35  
kNN\_clustering, 36  
louvainClustering, 37  
macrophage\_KEGG\_example, 38  
macrophage\_Reactome\_example, 38  
macrophage\_topGO\_example, 39  
macrophage\_topGO\_example\_small, 39  
markovClustering, 40  
Matrix::Matrix(), 5, 10, 15–17, 20, 21, 23, 27–30, 32, 34, 36, 37, 40, 41, 45–47  
pamClustering, 40  
pMMlocal, 41  
ppi\_macrophage\_topGO\_example\_small, 42  
prepareGenesetData, 43  
prepareGenesetData(), 15  
sample\_geneset, 44  
sample\_geneset\_broken, 44  
sample\_geneset\_empty, 44  
sample\_geneset\_small, 45  
scaleGO, 45  
scores\_macrophage\_topGO\_example\_small, 46  
seedFinding, 47  
shiny::Progress(), 27–30, 32, 34



stats::hclust(), [16](#)

STRINGdb, [22](#), [25](#), [31–33](#)

visNetwork::visIgraph(), [9](#), [10](#), [22](#)

wordcloud2::wordcloud2(), [18](#)