Package: pathfindR (via r-universe)

July 2, 2026

Type Package

Title Enrichment Analysis Utilizing Active Subnetworks

Version 3.0.2

Maintainer Ege Ulgen <egeulgen@gmail.com>

Description Enrichment analysis enables researchers to uncover mechanisms underlying a phenotype. However, conventional methods for enrichment analysis do not take into account protein-protein interaction information, resulting in incomplete conclusions. 'pathfindR' is a tool for enrichment analysis utilizing active subnetworks. The main function identifies active subnetworks in a protein-protein interaction network using a user-provided list of genes and associated p values. It then performs enrichment analyses on the identified subnetworks, identifying enriched terms (i.e. pathways or, more broadly, gene sets) that possibly underlie the phenotype of interest. 'pathfindR' also offers functionalities to cluster the enriched terms and identify representative terms in each cluster, to score the enriched terms per sample and to visualize analysis results. The enrichment, clustering and other methods implemented in 'pathfindR' are described in detail in Ulgen E, Ozisik O, Sezerman OU. 2019. 'pathfindR': An R Package for Comprehensive Identification of Enriched Pathways in Omics Data Through Active Subnetworks. Front. Genet. <doi:10.3389/fgene.2019.00858>.

License MIT + file LICENSE

URL https://egeulgen.github.io/pathfindR/, https://github.com/egeulgen/pathfindR

BugReports https://github.com/egeulgen/pathfindR/issues

Encoding UTF-8

LazyData true

Imports DBI, AnnotationDbi, doParallel, foreach, rmarkdown, ggplot2, ggraph, ggupset, ggnewscale, fpc, ggkegg (>= 1.4.0), grDevices, httr, igraph, R.utils, msigdbr (>= 24.1.0), knitr, Rcpp

Depends R (>= 4.3.0), pathfindR.data (>= 2.2.0)

Suggests org.Hs.eg.db, testthat (>= 3.0.0), covr, mockery

VignetteBuilder knitr

LinkingTo Rcpp

Config/testthat/parallel true

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Ege Ulgen [cre, cph] (ORCID:
<https://orcid.org/0000-0003-2090-3621>), Ozan Ozisik [aut]
(ORCID: <https://orcid.org/0000-0001-5980-8002>)

Config/pak/sysreqs zlib1g-dev

Repository https://cran.r-universe.dev

Date/Publication 2026-07-01 22:50:02 UTC

RemoteUrl https://github.com/cran/pathfindR

RemoteRef HEAD

RemoteSha cbbd36331254991e8eba8ad4dab89702ff704d67

Contents

.find_components_named *Find the connected components among a set of "on" nodes*

Description

Returns one group of node names per connected component of the subgraph induced by `on_names` (isolated nodes form singleton components).

Usage

```
.find_components_named(network, on_names)
```

Arguments

`network`	A network from `build_network()`.
`on_names`	Character vector of node names that are switched on.

Value

A list of character vectors, one per component.

.find_subnetworks *Find the scored subnetworks among a set of "on" nodes*

Description

Find the scored subnetworks among a set of "on" nodes

Usage

```
.find_subnetworks(network, sc, on_names)
```

Arguments

`network`	A network from `build_network()`.
`sc`	A score context.
`on_names`	Character vector of node names that are switched on.

Value

A list of subnetwork objects.

.ga_compare *Compare two individuals rank-by-rank*

Description

Compares the score-sorted subnetwork scores position by position; the first strict difference decides. If one score vector is a prefix of the other, the individual with more subnetworks wins.

Usage

```
.ga_compare(a, b)
```

Arguments

a, b	GA individuals.

Value

1 if a is better, -1 if b is better, 0 if they are equivalent.

.ga_crossover_mutation *Crossover and mutation of two parent genomes*

Description

With probability ga_crossover_rate the parents are recombined (uniform crossover by default); otherwise the children are empty genomes. Mutation flips each bit with probability ga_mutation_rate.

Usage

```
.ga_crossover_mutation(p1, p2, params, crossover_type = "UNIFORM")
```

Arguments

p1, p2	Parent genomes (logical vectors).
params	A list of run parameters.
crossover_type	One of "UNIFORM", "SINGLEPOINT" or "MULTIPOINT".

Value

A list of two child genomes (logical vectors).

.ga_make_individual *Build a GA individual from a logical genome*

Description

Computes the connected subnetworks induced by the on-nodes and stores their scores (sorted descending) for fast fitness comparison. The scores are obtained with the C++ component scorer over the precomputed CSR adjacency, which is the only thing the fitness comparison (`.ga_compare` / `.ga_sort_desc`) ever reads. The node membership of a genome is not materialized here — it is reconstructed once, for the best individual, at the end of `.genetic_algorithm()`.

Usage

```
.ga_make_individual(rep_logical, csr_offsets, csr_nbrs, z_vec, means, stds)
```

Arguments

`rep_logical`	A logical vector aligned to `network$nodes` (`TRUE` = node on). May be empty.
`csr_offsets, csr_nbrs`	CSR adjacency from `build_network()`.
`z_vec`	Per-node z-scores aligned to the node order.
`means, stds`	Score-context calibration vectors.

Value

A list with elements `rep` (the genome) and `scores` (the descending component scores).

.ga_pick *Pick one index by rank-proportional (roulette) selection*

Description

Pick one index by rank-proportional (roulette) selection

Usage

```
.ga_pick(weights, total)
```

Arguments

`weights`	Numeric weights aligned to the (sorted) population.
`total`	Sum of `weights`.

Value

An integer index into the population.

.ga_score *Fitness of an individual*

Description

The score of its highest scoring subnetwork, or `0` when it has none.

Usage

```
.ga_score(ind)
```

Arguments

`ind`	A GA individual from `.ga_make_individual()`.

Value

A numeric score.

.ga_sort_desc *Sort a population from best to worst*

Description

Implements the same ordering as `.ga_compare()` in a vectorised way: score vectors are laid out in a matrix padded with `-Inf`, then ordered lexicographically (descending), with the number of subnetworks as a final tie-break so that more subnetworks ranks higher.

Usage

```
.ga_sort_desc(pop)
```

Arguments

`pop`	A list of GA individuals.

Value

The population reordered, best individual first.

.genetic_algorithm *Run the genetic-algorithm active subnetwork search*

Description

The initial population is seeded randomly (each gene switched on with probability `params$gene_init_prob`); when `params$start_with_all_positives` is set, one individual containing all positive-z-score nodes is added. The population then evolves by rank-based selection, uniform crossover and optional mutation. Every ten generations the worst 10% of the population is replaced with fresh random individuals, and the previous best individual is preserved. The search stops after `params$ga_iterations` generations or once the best individual is unchanged for 50 generations.

Usage

```
.genetic_algorithm(network, sc, params, verbose = FALSE)
```

Arguments

`network`	A network from `build_network()`.
`sc`	A score context from `build_score_context()`.
`params`	A list of run parameters.
`verbose`	Logical; emit progress messages.

Value

A list of subnetwork objects from the best individual found.

.greedy_search *Run the greedy active subnetwork search*

Description

Run the greedy active subnetwork search

Usage

```
.greedy_search(network, score_context, params, verbose = FALSE)
```

Arguments

`network`	A network from `build_network()`.
`score_context`	A score context from `build_score_context()`.
`params`	A list of run parameters.
`verbose`	Logical; emit progress messages.

Value

A list of subnetwork objects.

.make_subnetwork *Create a subnetwork object from a vector of node names*

Description

Create a subnetwork object from a vector of node names

Usage

```
.make_subnetwork(sc, nodes)
```

Arguments

sc	A score context.
nodes	Character vector of node names.

Value

A list with elements nodes, zsum and score.

.parse_experiment *Parse the experiment input into a clean gene / p-value data frame*

Description

Accepts a data frame. If columns named gene and pvalue (case-insensitive) exist they are used, otherwise the first two columns are taken as gene and p-value respectively. Gene names are upper-cased.

Usage

```
.parse_experiment(experiment)
```

Arguments

experiment	A data frame of gene / p-value pairs.

Value

A data frame with character column gene and numeric column pvalue.

.score_subnetwork *Score a subnetwork from its size and z-score sum*

Description

Single-node subnetworks always score 0. Otherwise the raw score is `zsum / sqrt(n)`, optionally calibrated against the Monte-Carlo distribution and optionally penalized for size.

Usage

```
.score_subnetwork(sc, n, zsum, normalize)
```

Arguments

`sc`	A score context from `build_score_context()`.
`n`	Number of nodes in the subnetwork.
`zsum`	Sum of the z-scores of the subnetwork nodes.
`normalize`	Logical; whether to calibrate the score.

Value

A numeric score.

.simulated_annealing *Run the simulated annealing active subnetwork search*

Description

The candidate solution starts either with all positive-z-score nodes on (when `params$start_with_all_positives`) or with each node switched on independently with probability `params$gene_init_prob`. A random node is toggled each step and the resulting subnetworks are compared rank-by-rank against the current ones, accepting the change with a temperature-dependent probability. The temperature decays geometrically from `sa_initial_temp` to `sa_final_temp` over `sa_iterations` steps.

Usage

```
.simulated_annealing(network, sc, params, verbose = FALSE)
```

Arguments

`network`	A network from `build_network()` (provides the CSR adjacency `csr_offsets` / `csr_nbrs`).
`sc`	A score context from `build_score_context()`.
`params`	A list of run parameters.
`verbose`	Logical; emit progress messages.

Details

The search loop runs in C++ (`run_simulated_annealing`) and reproduces the Java reference exactly: the same `java.util.Random` stream (seeded with `params$seed`), the same initial-state draw order (node order), the same `nextInt`-based node toggling, and the same acceptance walk over the score-ranked subnetworks (advance to the next pair when a worse move passes its Boltzmann draw; accept on a >0.001 improvement; otherwise reject).

Value

A list of subnetwork objects (all connected components of the final solution, sorted by score descending).

`.sort_subnetworks_desc` *Sort a list of subnetwork objects by score, descending*

Description

Sort a list of subnetwork objects by score, descending

Usage

```
.sort_subnetworks_desc(subs)
```

Arguments

`subs`	A list of subnetwork objects.

Value

The list reordered so the highest scoring subnetwork is first.

`active_snw_enrichment_wrapper` *Wrapper for Active Subnetwork Search + Enrichment over Single/Multiple Iteration(s)*

Description

Wrapper for Active Subnetwork Search + Enrichment over Single/Multiple Iteration(s)

Usage

```
active_snw_enrichment_wrapper(
  input_processed,
  pin_path,
  gset_list,
  enrichment_threshold,
  list_active_snw_genes,
  adj_method = "bonferroni",
  search_method = "GR",
  disable_parallel = FALSE,
  start_with_all_positives = FALSE,
  iterations = 10,
  n_processes = NULL,
  score_quan_thr = 0.8,
  sig_gene_thr = 0.02,
  sa_initial_temp = 1,
  sa_final_temp = 0.01,
  sa_iterations = 10000,
  ga_population_size = 400,
  ga_iterations = 200,
  ga_crossover_rate = 1,
  ga_mutation_rate = 0,
  gr_max_depth = 1,
  gr_search_depth = 1,
  gr_overlap_threshold = 0.5,
  gr_subnetwork_num = 1000,
  verbose = FALSE
)
```

Arguments

input_processed	processed input data frame
pin_path	path/to/PIN/file
gset_list	list for gene sets.
enrichment_threshold	adjusted-p value threshold used when filtering enrichment results (default = 0.05)
list_active_snw_genes	boolean value indicating whether or not to report the non-significant active subnetwork genes for the active subnetwork which was enriched for the given term with the lowest p value (default = FALSE)
adj_method	correction method to be used for adjusting p-values. (default = 'bonferroni')
search_method	algorithm to use when performing active subnetwork search. Options are greedy search (GR), simulated annealing (SA) or genetic algorithm (GA) for the search (default = 'GR').

`disable_parallel`
: boolean to indicate whether to disable parallel runs via `foreach` (default = FALSE)

`start_with_all_positives`
: if TRUE: in GA, adds an individual with all positive nodes. In SA, initializes candidate solution with all positive nodes. (default = FALSE)

`iterations`
: number of iterations for active subnetwork search and enrichment analyses (Default = 10)

`n_processes`
: optional argument for specifying the number of processes used by foreach. If not specified, the function determines this automatically (Default == NULL. Gets set to 1 for Genetic Algorithm)

`score_quan_thr`
: active subnetwork score quantile threshold. Must be between 0 and 1 or set to -1 for not filtering. (Default = 0.8)

`sig_gene_thr`
: threshold for the minimum proportion of significant genes in the subnetwork (Default = 0.02) If the number of genes to use as threshold is calculated to be < 2 (e.g. 50 signif. genes x 0.01 = 0.5), the threshold number is set to 2

`sa_initial_temp`
: Initial temperature for SA (default = 1.0)

`sa_final_temp`
: Final temperature for SA (default = 0.01)

`sa_iterations`
: Iteration number for SA (default = 10000)

`ga_population_size`
: Population size for GA (default = 400)

`ga_iterations`
: Iteration number for GA (default = 200)

`ga_crossover_rate`
: Applies crossover with the given probability in GA (default = 1, i.e. always perform crossover)

`ga_mutation_rate`
: For GA, applies mutation with given mutation rate (default = 0, i.e. mutation off)

`gr_max_depth`
: Sets max depth in greedy search, 0 for no limit (default = 1)

`gr_search_depth`
: Search depth in greedy search (default = 1)

`gr_overlap_threshold`
: Overlap threshold for results of greedy search (default = 0.5)

`gr_subnetwork_num`
: Number of subnetworks to be presented in the results (default = 1000)

`verbose`
: boolean value indicating whether to print messages (default=FALSE)

Value

Data frame of combined pathfindR enrichment results

`active_subnetwork_search` *Active subnetwork search*

Description

Searches a molecular interaction network for connected subnetworks of genes that are jointly enriched for low experimental p-values ("active modules"). Gene p-values are converted to z-scores, subnetwork scores are calibrated against a Monte-Carlo background, and one of three search strategies is used to find high-scoring connected subnetworks.

Usage

```
active_subnetwork_search(
  network,
  score_context,
  method = c("GR", "SA", "GA"),
  params,
  verbose = FALSE
)
```

Arguments

`network`	A network list as returned by `build_network()`.
`score_context`	A score context list as returned by `build_score_context()`.
`method`	Search strategy: `"GR"` (greedy, the default), `"SA"` (simulated annealing) or `"GA"` (genetic algorithm).
`params`	A fully-formed params list controlling the search, as built by `get_active_subnetworks()`.
`verbose`	Logical; emit progress messages.

Value

A list of subnetwork objects sorted by score descending, each with elements `nodes` (character vector of gene names) and `score`. The list is empty when no positive-scoring subnetwork is found.

See Also

`get_active_subnetworks` which builds `network`, `score_context` and `params` and calls this function.

Examples

```
## Not run:
# Write a minimal SIF file
sif <- data.frame(
  source = c("A", "A", "B", "C", "D"),
  type   = "interacts",
```

```
  target = c("B", "C", "C", "D", "E")
)
sif_path <- tempfile(fileext = ".sif")
write.table(sif, sif_path,
  sep = "\t", row.names = FALSE, col.names = FALSE,
  quote = FALSE
)

experiment <- data.frame(
  gene   = c("A", "B", "C", "D", "E"),
  pvalue = c(0.001, 0.002, 0.001, 0.5, 0.6)
)

params <- list(
  start_with_all_positives = FALSE,
  gene_init_prob           = 0.1,
  p_for_nonsignificant     = 0.5,
  sa_initial_temp          = 1.0,
  sa_final_temp            = 0.01,
  sa_iterations            = 10000L,
  ga_population_size       = 400L,
  ga_iterations            = 200L,
  ga_crossover_rate        = 1,
  ga_mutation_rate         = 0,
  gr_max_depth             = 1L,
  gr_search_depth          = 1L,
  gr_overlap_threshold     = 0.5,
  gr_subnetwork_num        = 1000,
  seed                     = 1234L
)

network <- build_network(sif_path)
sc <- build_score_context(network, experiment, params)
snws <- active_subnetwork_search(network, sc, method = "GR", params = params)

## End(Not run)
```

`annotate_term_genes` *Annotate the Affected Genes in the Provided Enriched Terms*

Description

Function to annotate the involved affected (input) genes in each term.

Usage

```
annotate_term_genes(
  result_df,
  input_processed,
```

```
  genes_by_term = pathfindR.data::kegg_genes
)
```

Arguments

result_df	data frame of enrichment results. The only must-have column is 'ID'.
input_processed	input data processed via input_processing
genes_by_term	List that contains genes for each gene set. Names of this list are gene set IDs (default = kegg_genes)

Value

The original data frame with two additional columns:

Up_regulated the up-regulated genes in the input involved in the given term's gene set, comma-separated

Down_regulated the down-regulated genes in the input involved in the given term's gene set, comma-separated

Examples

```
example_gene_data <- example_pathfindR_input
colnames(example_gene_data) <- c("GENE", "CHANGE", "P_VALUE")

annotated_result <- annotate_term_genes(
  result_df = example_pathfindR_output,
  input_processed = example_gene_data
)
```

---

build_network *Build the undirected interaction network from a SIF file*

Description

Reads a Simple Interaction Format (SIF) file and converts it into an undirected igraph graph object. Following the Java reference's SIFReader, the column count is taken from the first line: a 2-column file uses columns 1 and 2 as the interacting nodes, a 3-column file uses columns 1 and 3 (the middle interaction-type column is ignored). Node names are upper-cased and self-interactions are discarded.

Usage

```
build_network(sif_path)
```

Arguments

sif_path	Character string specifying the path to the SIF file. The file should be whitespace/tab-delimited with 2 or 3 columns.

Details

To reproduce the Java implementation's greedy search bit-for-bit, this function also reconstructs two order-sensitive structures that the Java code derives from its HashMap/HashSet traversal: * `nodes`: the node order of Java's `networkNodeList` (`adjacency.keySet()` iteration order), via `java_node_order()`. * `nbr_idx`: per-node neighbour lists in Java's `HashSet` iteration order, via `java_neighbour_order()`. These orders drive both the Monte-Carlo calibration (which shuffles z-scores in node order) and the greedy expansion/removal, so matching them is what makes the R/C++ output align with the Java reference. The `igraph` object is retained for the SA / GA algorithms, whose component scoring is order-independent.

Value

A list with elements: `g` (an `igraph` graph), `nodes` (node names in Java `networkNodeList` order), `nbr` (named list of neighbour-name vectors in Java HashSet order), `nbr_idx` (list of 1-based neighbour-id vectors aligned to `nodes`, in Java HashSet order, ready for `run_greedy_search()`) and `name2id` (named integer vector mapping node name to its index in `nodes`) and `csr_offsets` / `csr_nbrs` (a compressed sparse-row, 0-based adjacency used by the SA / GA component scorer).

`build_score_context` *Build the score context*

Description

Computes per-node z-scores from the experiment p-values and the Monte-Carlo mean / standard deviation of the raw subnetwork score at every possible subnetwork size, used to calibrate (normalise) scores.

Usage

```
build_score_context(network, experiment, params)
```

Arguments

`network`	A network as returned by `build_network()`.
`experiment`	A data frame of gene / p-value pairs.
`params`	A list of run parameters. `params$seed` must match the Java `-seedForRandom` value (Java default 1234) for identical calibration.

Details

The Monte-Carlo step is an exact replica of the Java reference's `ScoreCalculations.calculateMeanAndStdForMonteCarl` it shuffles the z-score vector 2000 times with a `java.util.Random`-equivalent generator and Fisher-Yates shuffle (see `get_java_mc_calibration()` in '`score_calculation_utils.cpp`'). Because `Collections.shuffle` starts from `networkNodeList` order, the z-score vector MUST be in Java node order; `network$nodes` (from `build_network()`) provides exactly that, so `z` is built over it directly. With the matching seed this reproduces the Java means/stds to floating-point precision, which is what aligns the calibrated scores and rankings with the Java output.

P-values are clamped to a safe range, the smallest p-value is kept when a gene appears more than once, and network nodes without a p-value are given `params$p_for_nonsignificant`.

Value

A list with elements `z` (named z-score vector), `means` and `stds` (numeric vectors indexed by subnetwork size), and `nodes`.

`cluster_enriched_terms` *Cluster Enriched Terms*

Description

Cluster Enriched Terms

Usage

```
cluster_enriched_terms(
  enrichment_res,
  method = "hierarchical",
  plot_clusters_graph = TRUE,
  use_description = FALSE,
  use_active_snw_genes = FALSE,
  ...
)
```

Arguments

`enrichment_res`	data frame of pathfindR enrichment results. Must-have columns are 'Term_Description' (if `use_description = TRUE`) or 'ID' (if `use_description = FALSE`), 'Down_regulated', and 'Up_regulated'. If `use_active_snw_genes = TRUE`, 'non_Signif_Snw_Genes' must also be provided.
`method`	Either 'hierarchical' or 'fuzzy'. Details of clustering are provided in the corresponding functions `hierarchical_term_clustering`, and `fuzzy_term_clustering`
`plot_clusters_graph`	boolean value indicate whether or not to plot the graph diagram of clustering results (default = TRUE)
`use_description`	Boolean argument to indicate whether term descriptions (in the 'Term_Description' column) should be used. (default = `FALSE`)
`use_active_snw_genes`	boolean to indicate whether or not to use non-input active subnetwork genes in the calculation of kappa statistics (default = FALSE, i.e. only use affected genes)
`...`	additional arguments for `hierarchical_term_clustering`, `fuzzy_term_clustering` and `cluster_graph_vis`. See documentation of these functions for more details.

Value

a data frame of clustering results. For 'hierarchical', the cluster assignments (Cluster) and whether the term is representative of its cluster (Status) is added as columns. For 'fuzzy', terms that are in multiple clusters are provided for each cluster. The cluster assignments (Cluster) and whether the term is representative of its cluster (Status) is added as columns.

See Also

See hierarchical_term_clustering for hierarchical clustering of enriched terms. See fuzzy_term_clustering for fuzzy clustering of enriched terms. See cluster_graph_vis for graph visualization of clustering.

Examples

```
example_clustered <- cluster_enriched_terms(
  example_pathfindR_output[1:3, ],
  plot_clusters_graph = FALSE
)
example_clustered <- cluster_enriched_terms(
  example_pathfindR_output[1:3, ],
  method = "fuzzy", plot_clusters_graph = FALSE
)
```

---

cluster_graph_vis *Graph Visualization of Clustered Enriched Terms*

Description

Graph Visualization of Clustered Enriched Terms

Usage

```
cluster_graph_vis(
  clu_obj,
  kappa_mat,
  enrichment_res,
  kappa_threshold = 0.35,
  use_description = FALSE,
  vertex.label.cex = 0.7,
  vertex.size.scaling = 2.5
)
```

Arguments

clu_obj	clustering result (either a matrix obtained via hierarchical_term_clustering or fuzzy_term_clustering 'fuzzy_term_clustering' or a vector obtained via 'hierarchical_term_clustering')

kappa_mat — matrix of kappa statistics (output of create_kappa_matrix)

enrichment_res — data frame of pathfindR enrichment results. Must-have columns are 'Term_Description' (if use_description = TRUE) or 'ID' (if use_description = FALSE), 'Down_regulated', and 'Up_regulated'. If use_active_snw_genes = TRUE, 'non_Signif_Snw_Genes' must also be provided.

kappa_threshold — threshold for kappa statistics, defining strong relation (default = 0.35)

use_description — Boolean argument to indicate whether term descriptions (in the 'Term_Description' column) should be used. (default = FALSE)

vertex.label.cex — font size for vertex labels; it is interpreted as a multiplication factor of some device-dependent base font size (default = 0.7)

vertex.size.scaling — scaling factor for the node size (default = 2.5)

Value

Plots a graph diagram of clustering results. Each node is an enriched term from 'enrichment_res'. Size of node corresponds to -log(lowest_p). Thickness of the edges between nodes correspond to the kappa statistic between the two terms. Color of each node corresponds to distinct clusters. For fuzzy clustering, if a term is in multiple clusters, multiple colors are utilized.

Examples

```
## Not run:
cluster_graph_vis(clu_obj, kappa_mat, enrichment_res)

## End(Not run)
```

---

color_kegg_pathway *Color hsa KEGG pathway*

Description

Color hsa KEGG pathway

Usage

```
color_kegg_pathway(
  pw_id,
  change_vec,
  scale_vals = TRUE,
  node_cols = NULL,
  legend.position = "top"
)
```

Arguments

pw_id	hsa KEGG pathway id (e.g. hsa05012)
change_vec	vector of change values, names should be hsa KEGG gene ids
scale_vals	should change values be scaled? (default = TRUE)
node_cols	low, middle and high color values for coloring the pathway nodes (default = NULL). If node_cols=NULL, the low, middle and high color are set as 'green', 'gray' and 'red'. If all change values are 1e6 (in case no changes are supplied, this dummy value is assigned by input_processing), only one color ('#F38F18' if NULL) is used.
legend.position	the default position of legends ("none", "left", "right", "bottom", "top", "inside")

Value

a ggplot object containing the colored KEGG pathway diagram visualization

Examples

```
## Not run:
pw_id <- "hsa00010"
change_vec <- c(-2, 4, 6)
names(change_vec) <- c("hsa:2821", "hsa:226", "hsa:229")
result <- pathfindR:::color_kegg_pathway(pw_id, change_vec)

## End(Not run)
```

---

combine_pathfindR_results

Combine 2 pathfindR Results

Description

Combine 2 pathfindR Results

Usage

```
combine_pathfindR_results(result_A, result_B, plot_common = TRUE)
```

Arguments

result_A	data frame of first pathfindR enrichment results
result_B	data frame of second pathfindR enrichment results
plot_common	boolean to indicate whether or not to plot the term-gene graph of the common terms (default=TRUE)

Value

Data frame of combined pathfindR enrichment results. Columns are:

ID ID of the enriched term

Term_Description Description of the enriched term

Fold_Enrichment_A Fold enrichment value for the enriched term (Calculated using ONLY the input genes)

occurrence_A the number of iterations that the given term was found to enriched over all iterations

lowest_p_A the lowest adjusted-p value of the given term over all iterations

highest_p_A the highest adjusted-p value of the given term over all iterations

Up_regulated_A the up-regulated genes in the input involved in the given term's gene set, comma-separated

Down_regulated_A the down-regulated genes in the input involved in the given term's gene set, comma-separated

Fold_Enrichment_B Fold enrichment value for the enriched term (Calculated using ONLY the input genes)

occurrence_B the number of iterations that the given term was found to enriched over all iterations

lowest_p_B the lowest adjusted-p value of the given term over all iterations

highest_p_B the highest adjusted-p value of the given term over all iterations

Up_regulated_B the up-regulated genes in the input involved in the given term's gene set, comma-separated

Down_regulated_B the down-regulated genes in the input involved in the given term's gene set, comma-separated

combined_p the combined p value (via Fisher's method)

status whether the term is found in both analyses ('common'), found only in the first ('A only') or found only in the second ('B only)

By default, the function also displays the term-gene graph of the common terms

Examples

```
combined_results <- combine_pathfindR_results(example_pathfindR_output, example_comparison_output)
```

`combined_results_graph` *Combined Results Graph*

Description

Combined Results Graph

Usage

```
combined_results_graph(
  combined_df,
  selected_terms = "common",
  use_description = FALSE,
  layout = "stress",
  node_size = "num_genes"
)
```

Arguments

combined_df	Data frame of combined pathfindR enrichment results
selected_terms	the vector of selected terms for creating the graph (either IDs or term descriptions). If set to 'common', all of the common terms are used. (default = 'common')
use_description	Boolean argument to indicate whether term descriptions (in the 'Term_Description' column) should be used. (default: FALSE)
layout	The type of layout to create (see ggraph for details (default: 'stress')
node_size	Argument to indicate whether to use number of significant genes ('num_genes') or the -log10(lowest p value) ('p_val') for adjusting the term node sizes (default = 'num_genes')

Value

a ggraph object containing the combined term-gene graph. Each node corresponds to an enriched term (orange if common, different shades of blue otherwise), an up-regulated gene (green), a down-regulated gene (red) or a conflicting (i.e. up in one analysis, down in the other or vice versa) gene (gray). An edge between a term and a gene indicates that the given term involves the gene. Size of a term node is proportional to either the number of genes (if node_size = 'num_genes') or the -log10(lowest p value) (if node_size = 'p_val').

Examples

```
combined_results <- combine_pathfindR_results(
  example_pathfindR_output,
  example_comparison_output,
  plot_common = FALSE
)
g <- combined_results_graph(combined_results, selected_terms = sample(combined_results$ID, 3))
```

create_HTML_report *Create HTML Report of pathfindR Results*

Description

Create HTML Report of pathfindR Results

Usage

```
create_HTML_report(input, input_processed, final_res, dir_for_report)
```

Arguments

input	the input data that pathfindR uses. The input must be a data frame with three columns: 1. Gene Symbol (Gene Symbol) 2. Change value, e.g. log(fold change) (OPTIONAL) 3. p value, e.g. adjusted p value associated with differential expression
input_processed	processed input data frame
final_res	final pathfindR result data frame
dir_for_report	directory to render the report in

create_kappa_matrix *Create Kappa Statistics Matrix*

Description

Create Kappa Statistics Matrix

Usage

```
create_kappa_matrix(
  enrichment_res,
  use_description = FALSE,
  use_active_snw_genes = FALSE
)
```

Arguments

enrichment_res	data frame of pathfindR enrichment results. Must-have columns are 'Term_Description' (if use_description = TRUE) or 'ID' (if use_description = FALSE), 'Down_regulated', and 'Up_regulated'. If use_active_snw_genes = TRUE, 'non_Signif_Snw_Genes' must also be provided.
use_description	Boolean argument to indicate whether term descriptions (in the 'Term_Description' column) should be used. (default = FALSE)
use_active_snw_genes	boolean to indicate whether or not to use non-input active subnetwork genes in the calculation of kappa statistics (default = FALSE, i.e. only use affected genes)

Value

a matrix of kappa statistics between each term in the enrichment results.

Examples

```
sub_df <- example_pathfindR_output[1:3, ]
create_kappa_matrix(sub_df)
```

---

create_term_gene_graph *Create Term-Gene Graph*

Description

Create Term-Gene Graph

Usage

```
create_term_gene_graph(
  result_df,
  genes_df = NULL,
  order_by = "lowest_p",
  term_size = "num_genes",
  term_fill = NULL,
  num_terms = 10,
  use_description = FALSE,
  use_edge_weights = FALSE
)
```

Arguments

`result_df` A dataframe of pathfindR results that must contain the following columns:

Term_Description Description of the enriched term (necessary if `use_description = TRUE`)

ID ID of the enriched term (necessary if `use_description = FALSE`)

lowest_p the lowest adjusted-p value of the given term over all iterations

Up_regulated the up-regulated genes in the input involved in the given term's gene set, comma-separated

Down_regulated the down-regulated genes in the input involved in the given term's gene set, comma-separated

`genes_df` (optional) the input data that was used with `run_pathfindR` (default: `NULL`). It must be a data frame with at least 2 columns:

1. Gene.Symbol (required)
2. logFC (required)

`order_by` Argument to order the 'result_df', this influences the 'num_terms' displayed (default: `'lowest_p'`).

`term_size` Argument to indicate whether to use number of significant genes ('num_genes')

`term_fill` Argument to indicate by what column to fill the term nodes (e.g. `term_fill = "Fold_Enrichment"`) (default: `NULL`).

`num_terms` Number of top enriched terms to use while creating the graph. Set to `NULL` to use all enriched terms (default = 10, i.e. top 10 terms)

`use_description` Boolean argument to indicate whether term descriptions (in the 'Term_Description' column) should be used. (default: `FALSE`)

`use_edge_weights` Boolean argument to indicate whether genes are weighted by their term interactions, similar to an Up-Set plot but in graph context (default = `FALSE`). or the -log10(lowest p value) ('p_val') for adjusting the term node sizes (default: `'num_genes'`)

Details

This function constructs an igraph object from pathfindR output, creating a network that connects enriched biological terms to their involved genes. By default, the graph connects term nodes to up-regulated genes and down-regulated genes. The size of term nodes can be adjusted by either the number of significant genes ('term_size = 'num_genes'') or by the statistical significance ('term_size = 'p_val'', using -log10(lowest p value)).

When 'genes_df' is provided, gene nodes contain values and not mere up/down binary values, allowing visualization of expression direction and magnitude. When 'term_fill' is supplied, term nodes obtain values enabling simultaneous visualization of pathway enrichment strength.

Setting 'use_edge_weights = TRUE' highlights hub genes by weighting edges based on how many terms a gene participates in, similar to an Up-Set plot but in a graph context. The 'num_terms' parameter controls how many top enriched terms are included (default: top 10), and 'order_by' determines the ordering criterion for term selection. The resulting igraph object can be visualized using create_term_gene_plot.

Value

A igraph object

Examples

```
# Normal gene-term with up/down regulated genes
g <- create_term_gene_graph(
  result_df = example_pathfindR_output
)
g <- create_term_gene_graph(
  result_df = example_pathfindR_output,
  num_terms = 5
)
g <- create_term_gene_graph(
  result_df = example_pathfindR_output,
  term_size = "p_val"
)

# Coloring the term nodes
g <- create_term_gene_graph(
  result_df = example_pathfindR_output,
  term_fill = "Fold_Enrichment"
)

# Adding edge weights
g <- create_term_gene_graph(
  result_df = example_pathfindR_output,
  term_fill = "Fold_Enrichment",
  use_edge_weights = TRUE
)
```

---

create_term_gene_plot *Create Term-Gene Plot*

Description

Create Term-Gene Plot

Usage

```
create_term_gene_plot(
  graph,
  layout = "stress",
  gene_node_fill = c("#7E2795", "white", "#27AE60"),
  term_node_fill = c("#CCBB44", "white", "#4477AA"),
  gene_node_color = c("green", "red"),
  term_node_color = "#E5D7BF",
  term_fill_label = NULL,
  term_size_label = NULL
)
```

Arguments

graph	A igraph returned from create_term_gene_graph.
layout	The type of layout to create (see ggraph for details (default: 'stress')
gene_node_fill	A character vector to customize the fill gradient colors of the gene nodes when 'genes_df' is supplied, color order is in low -> mid -> high (default: c("#7E2795", "white", "#27AE60")).
term_node_fill	A character vector to customize the fill gradient colors of the term nodes when 'term_fill' is supplied, color order is in low -> mid -> high (default: c("#CCBB44", "white", "#4477AA")).
gene_node_color	A character vector to customize the fill gradient colors of the term nodes when 'genes_df' is not supplied, color order is in up -> down (default: c("green", "red")).
term_node_color	A character to customize the fill color of the terms when 'term_fill' is not specified (default: "#E5D7BF").
term_fill_label	A character to change the term node legend name (default: NULL).
term_size_label	A character to change the term node size legend name (default: NULL).

Details

This function creates a visualization of the term-gene graph (adapted from the Gene-Concept network visualization in the enrichplot package). It displays which input genes are involved in enriched biological terms, showing connections between genes and pathway/terms nodes. The graph facilitates investigation of multi-term relationships and identifies shared versus distinct genes across enriched terms.

Node coloring depends on the inputs provided to create_term_gene_graph:

- If 'genes_df' was NOT supplied: term nodes are beige ('term_node_color'), up-regulated genes are green, and down-regulated genes are red.
- If 'genes_df' WAS supplied: gene nodes are colored by logFC using a gradient ('gene_node_fill': default purple → white → green), and term nodes can be colored by 'term_fill' values (default yellow → white → blue) if 'term_fill' was provided.

Term node size reflects either the number of associated genes ('term_size = 'num_genes'') or statistical significance ('term_size = 'p_val''). When 'use_edge_weights = TRUE' was set in 'create_term_gene_graph', edge widths represent hub gene importance (genes appearing in multiple terms). The layout can be customized via the 'layout' parameter (default: "stress"), and legends automatically reflect the applied coloring schemes.

Value

A ggraph object

Examples

```
# Normal gene-term with up/down regulated genes
g <- create_term_gene_graph(
  result_df = example_pathfindR_output
)
plt <- create_term_gene_plot(g)
```

`enrichment` *Perform Enrichment Analysis for a Single Gene Set*

Description

Perform Enrichment Analysis for a Single Gene Set

Usage

```
enrichment(
  input_genes,
  genes_by_term = pathfindR.data::kegg_genes,
  term_descriptions = pathfindR.data::kegg_descriptions,
  adj_method = "bonferroni",
  enrichment_threshold = 0.05,
  sig_genes_vec,
  background_genes
)
```

Arguments

`input_genes`	The set of gene symbols to be used for enrichment analysis. In the scope of this package, these are genes that were identified for an active subnetwork
`genes_by_term`	List that contains genes for each gene set. Names of this list are gene set IDs (default = kegg_genes)
`term_descriptions`	Vector that contains term descriptions for the gene sets. Names of this vector are gene set IDs (default = kegg_descriptions)
`adj_method`	correction method to be used for adjusting p-values. (default = 'bonferroni')
`enrichment_threshold`	adjusted-p value threshold used when filtering enrichment results (default = 0.05)
`sig_genes_vec`	vector of significant gene symbols. In the scope of this package, these are the input genes that were used for active subnetwork search
`background_genes`	vector of background genes. In the scope of this package, the background genes are taken as all genes in the PIN (see `enrichment_analyses`)

Value

A data frame that contains enrichment results

See Also

p.adjust for adjustment of p values. See run_pathfindR for the wrapper function of the pathfindR workflow. hyperg_test for the details on hypergeometric distribution-based hypothesis testing.

Examples

```
enrichment(
  input_genes = c("PER1", "PER2", "CRY1", "CREB1"),
  sig_genes_vec = "PER1",
  background_genes = unlist(pathfindR.data::kegg_genes)
)
```

---

enrichment_analyses *Perform Enrichment Analyses on the Input Subnetworks*

Description

Perform Enrichment Analyses on the Input Subnetworks

Usage

```
enrichment_analyses(
  snws,
  sig_genes_vec,
  pin_name_path = "Biogrid",
  genes_by_term = pathfindR.data::kegg_genes,
  term_descriptions = pathfindR.data::kegg_descriptions,
  adj_method = "bonferroni",
  enrichment_threshold = 0.05,
  list_active_snw_genes = FALSE
)
```

Arguments

snws	a list of subnetwork genes (i.e., vectors of genes for each subnetwork)
sig_genes_vec	vector of significant gene symbols. In the scope of this package, these are the input genes that were used for active subnetwork search
pin_name_path	Name of the chosen PIN or absolute/path/to/PIN.sif. If PIN name, must be one of c('Biogrid', 'STRING', 'GeneMania', 'IntAct', 'KEGG', 'mmu_STRING'). If path/to/PIN.sif, the file must comply with the PIN specifications. (Default = 'Biogrid')
genes_by_term	List that contains genes for each gene set. Names of this list are gene set IDs (default = kegg_genes)

`term_descriptions`
: Vector that contains term descriptions for the gene sets. Names of this vector are gene set IDs (default = kegg_descriptions)

`adj_method`
: correction method to be used for adjusting p-values. (default = 'bonferroni')

`enrichment_threshold`
: adjusted-p value threshold used when filtering enrichment results (default = 0.05)

`list_active_snw_genes`
: boolean value indicating whether or not to report the non-significant active subnetwork genes for the active subnetwork which was enriched for the given term with the lowest p value (default = `FALSE`)

Value

a dataframe of combined enrichment results. Columns are:

ID ID of the enriched term

Term_Description Description of the enriched term

Fold_Enrichment Fold enrichment value for the enriched term

p_value p value of enrichment

adj_p adjusted p value of enrichment

support the support (proportion of active subnetworks leading to enrichment over all subnetworks) for the gene set

non_Signif_Snw_Genes (OPTIONAL) the non-significant active subnetwork genes, comma-separated

See Also

`enrichment` for the enrichment analysis for a single gene set

Examples

```
enr_res <- enrichment_analyses(
  snws = example_active_snws$subnetworks[1:2],
  sig_genes_vec = example_pathfindR_input$Gene.symbol[1:25],
  pin_name_path = "KEGG"
)
```

---

enrichment_chart *Create Bubble Chart of Enrichment Results*

Description

This function is used to create a ggplot2 bubble chart displaying the enrichment results.

Usage

```
enrichment_chart(
  result_df,
  top_terms = 10,
  plot_by_cluster = FALSE,
  num_bubbles = 4,
  even_breaks = TRUE,
  order_by = "lowest_p"
)
```

Arguments

result_df	a data frame that must contain the following columns: **Term_Description** Description of the enriched term **Fold_Enrichment** Fold enrichment value for the enriched term **lowest_p** the lowest adjusted-p value of the given term over all iterations **Up_regulated** the up-regulated genes in the input involved in the given term's gene set, comma-separated **Down_regulated** the down-regulated genes in the input involved in the given term's gene set, comma-separated **Cluster(OPTIONAL)** the cluster to which the enriched term is assigned
top_terms	number of top terms (according to the 'lowest_p' column) to plot (default = 10). If `plot_by_cluster = TRUE`, selects the top `top_terms` terms per each cluster. Set `top_terms = NULL` to plot for all terms.If the total number of terms is less than `top_terms`, all terms are plotted.
plot_by_cluster	boolean value indicating whether or not to group the enriched terms by cluster (works if `result_df` contains a 'Cluster' column).
num_bubbles	number of sizes displayed in the legend # genes (Default = 4)
even_breaks	whether or not to set even breaks for the number of sizes displayed in the legend # genes. If TRUE (default), sets equal breaks and the number of displayed bubbles may be different than the number set by `num_bubbles`. If the exact number set by `num_bubbles` is required, set this argument to FALSE
order_by	the order and coloring of the dots (default: `'lowest_p'`).

Value

a ggplot2 object containing the bubble chart. The x-axis corresponds to fold enrichment values while the y-axis indicates the enriched terms. Size of the bubble indicates the number of significant genes in the given enriched term. Color indicates the -log10(lowest-p) value. The closer the color is to red, the more significant the enrichment is. Optionally, if 'Cluster' is a column of `result_df` and `plot_by_cluster == TRUE`, the enriched terms are grouped by clusters.

Examples

```
g <- enrichment_chart(example_pathfindR_output)
```

`example_unfiltered_snws` *Example Unfiltered Active Subnetworks*

Description

List containing unfiltered active subnetworks returned by `active_subnetwork_search`

Usage

```
example_unfiltered_snws
```

Format

A list with 1000 elements. Each containing a named list:

nodes active subnetwork nodes

score score of the active subnetwork

`fetch_gene_sets` *Fetch Gene Set Objects*

Description

Function for obtaining the gene sets per term and the term descriptions to be used for enrichment analysis.

Usage

```
fetch_gene_sets(
  gene_sets = "KEGG",
  min_gset_size = 10,
  max_gset_size = 300,
  custom_genes = NULL,
  custom_descriptions = NULL
)
```

Arguments

`gene_sets`	Name of the gene sets to be used for enrichment analysis. Available gene sets are 'KEGG', 'Reactome', 'BioCarta', 'GO-All', 'GO-BP', 'GO-CC', 'GO-MF', 'cell_markers', 'mmu_KEGG' or 'Custom'. If 'Custom', the arguments `custom_genes` and `custom_descriptions` must be specified. (Default = 'KEGG')
`min_gset_size`	minimum number of genes a term must contain (default = 10)
`max_gset_size`	maximum number of genes a term must contain (default = 300)

custom_genes a list containing the genes involved in each custom term. Each element is a vector of gene symbols located in the given custom term. Names should correspond to the IDs of the custom terms.

custom_descriptions
A vector containing the descriptions for each custom term. Names of the vector should correspond to the IDs of the custom terms.

Value

a list containing 2 elements

genes_by_term list of vectors of genes contained in each term

term_descriptions vector of descriptions per each term

Examples

```
KEGG_gset <- fetch_gene_sets()
GO_MF_gset <- fetch_gene_sets("GO-MF", min_gset_size = 20, max_gset_size = 100)
```

---

filter_active_subnetworks

Parse Active Subnetwork Search Output File and Filter the Subnetworks

---

Description

Parse Active Subnetwork Search Output File and Filter the Subnetworks

Usage

```
filter_active_subnetworks(
  active_snws,
  sig_genes_vec,
  score_quan_thr = 0.8,
  sig_gene_thr = 0.02
)
```

Arguments

active_snws active subnetwork search results. A list containing input `subnetworks` (nodes) and `scores` (score).

sig_genes_vec vector of significant gene symbols. In the scope of this package, these are the input genes that were used for active subnetwork search

score_quan_thr active subnetwork score quantile threshold. Must be between 0 and 1 or set to -1 for not filtering. (Default = 0.8)

sig_gene_thr threshold for the minimum proportion of significant genes in the subnetwork (Default = 0.02) If the number of genes to use as threshold is calculated to be < 2 (e.g. 50 signif. genes x 0.01 = 0.5), the threshold number is set to 2

Value

A list containing `subnetworks`: a list of of genes in every active subnetwork that has a score greater than the `score_quan_thrth` quantile and that contains at least `sig_gene_thr` of significant genes and `scores` the score of each filtered active subnetwork

See Also

See `run_pathfindR` for the wrapper function of the pathfindR enrichment workflow

Examples

```
filtered <- filter_active_subnetworks(
  active_snws = example_unfiltered_snws,
  sig_genes_vec = example_pathfindR_input$Gene.symbol
)
```

`fuzzy_term_clustering` *Heuristic Fuzzy Multiple-linkage Partitioning of Enriched Terms*

Description

Heuristic Fuzzy Multiple-linkage Partitioning of Enriched Terms

Usage

```
fuzzy_term_clustering(
  kappa_mat,
  enrichment_res,
  kappa_threshold = 0.35,
  use_description = FALSE
)
```

Arguments

`kappa_mat`	matrix of kappa statistics (output of `create_kappa_matrix`)
`enrichment_res`	data frame of pathfindR enrichment results. Must-have columns are 'Term_Description' (if `use_description = TRUE`) or 'ID' (if `use_description = FALSE`), 'Down_regulated', and 'Up_regulated'. If `use_active_snw_genes = TRUE`, 'non_Signif_Snw_Genes' must also be provided.
`kappa_threshold`	threshold for kappa statistics, defining strong relation (default = 0.35)
`use_description`	Boolean argument to indicate whether term descriptions (in the 'Term_Description' column) should be used. (default = `FALSE`)

Details

The fuzzy clustering algorithm was implemented based on: Huang DW, Sherman BT, Tan Q, et al. The DAVID Gene Functional Classification Tool: a novel biological module-centric algorithm to functionally analyze large gene lists. Genome Biol. 2007;8(9):R183.

Value

a boolean matrix of cluster assignments. Each row corresponds to an enriched term, each column corresponds to a cluster.

Examples

```
## Not run:
fuzzy_term_clustering(kappa_mat, enrichment_res)
fuzzy_term_clustering(kappa_mat, enrichment_res, kappa_threshold = 0.45)

## End(Not run)
```

---

get_active_subnetworks

Get Active Subnetworks

Description

Performs active subnetwork search and filters identified active subnetworks before returning final subnetworks.

Usage

```
get_active_subnetworks(
  significant_genes,
  network,
  score_context,
  score_quan_thr = 0.8,
  sig_gene_thr = 0.02,
  search_method = "GR",
  seed_for_stochastic_methods = 1234,
  verbose = FALSE,
  start_with_all_positives = FALSE,
  gene_init_prob = 0.1,
  sa_initial_temp = 1,
  sa_final_temp = 0.01,
  sa_iterations = 10000,
  ga_population_size = 400,
  ga_iterations = 200,
  ga_crossover_rate = 1,
  ga_mutation_rate = 0,
```

```
  gr_max_depth = 1,
  gr_search_depth = 1,
  gr_overlap_threshold = 0.5,
  gr_subnetwork_num = 1000
)
```

Arguments

significant_genes	vector of significant genes for the experiment
network	Prebuilt network object as returned by `build_network()`, built once by `active_snw_enrichment_wrapp` and passed to every iteration to avoid redundant PIN file I/O.
score_context	Prebuilt score context as returned by `build_score_context()`, built once by `active_snw_enrichment_wrapper` and passed to every iteration to avoid redundant Monte-Carlo calibration.
score_quan_thr	active subnetwork score quantile threshold. Must be between 0 and 1 or set to -1 for not filtering. (Default = 0.8)
sig_gene_thr	threshold for the minimum proportion of significant genes in the subnetwork (Default = 0.02) If the number of genes to use as threshold is calculated to be < 2 (e.g. 50 signif. genes x 0.01 = 0.5), the threshold number is set to 2
search_method	algorithm to use when performing active subnetwork search. Options are greedy search (GR), simulated annealing (SA) or genetic algorithm (GA) for the search (default = 'GR').
seed_for_stochastic_methods	seed for reproducibility while running active subnetwork search
verbose	boolean value indicating whether to print messages (default=FALSE)
start_with_all_positives	if TRUE: in GA, adds an individual with all positive nodes. In SA, initializes candidate solution with all positive nodes. (default = FALSE)
gene_init_prob	For SA and GA, probability of adding a gene in initial solution (default = 0.1)
sa_initial_temp	Initial temperature for SA (default = 1.0)
sa_final_temp	Final temperature for SA (default = 0.01)
sa_iterations	Iteration number for SA (default = 10000)
ga_population_size	Population size for GA (default = 400)
ga_iterations	Iteration number for GA (default = 200)
ga_crossover_rate	Applies crossover with the given probability in GA (default = 1, i.e. always perform crossover)
ga_mutation_rate	For GA, applies mutation with given mutation rate (default = 0, i.e. mutation off)
gr_max_depth	Sets max depth in greedy search, 0 for no limit (default = 1)

gr_search_depth
: Search depth in greedy search (default = 1)

gr_overlap_threshold
: Overlap threshold for results of greedy search (default = 0.5)

gr_subnetwork_num
: Number of subnetworks to be presented in the results (default = 1000)

Value

A list of genes in every identified active subnetwork that has a score greater than the 'score_quan_thr'th quantile and that has at least 'sig_gene_thr' affected genes.

Examples

```
experiment_df <- example_pathfindR_input[1:15, c(1, 3)]
colnames(experiment_df) <- c("gene", "pvalue")
pin_path <- return_pin_path("KEGG")
network <- build_network(pin_path)
score_context <- build_score_context(
  network,
  experiment_df,
  list(p_for_nonsignificant = 0.5, seed = 1234L)
)
GR_snws <- get_active_subnetworks(
  significant_genes = experiment_df$gene,
  network = network,
  score_context = score_context
)
```

---

get_biogrid_pin *Retrieve the Requested Release of Organism-specific BioGRID PIN*

Description

Retrieve the Requested Release of Organism-specific BioGRID PIN

Usage

```
get_biogrid_pin(org = "Homo_sapiens", path2pin, release = "latest")
```

Arguments

org	organism name. BioGRID naming requires underscores for spaces so 'Homo sapiens' becomes 'Homo_sapiens', 'Mus musculus' becomes 'Mus_musculus' etc. See https://wiki.thebiogrid.org/doku.php/statistics for a full list of available organisms (default = 'Homo_sapiens')
path2pin	the path of the file to save the PIN data. By default, the PIN data is saved in a temporary file
release	the requested BioGRID release (default = 'latest')

Value

the path of the file in which the PIN data was saved. If `path2pin` was not supplied by the user, the PIN data is saved in a temporary file

---

`get_gene_sets_list` *Retrieve Organism-specific Gene Sets List*

Description

Retrieve Organism-specific Gene Sets List

Usage

```
get_gene_sets_list(
  source = "KEGG",
  org_code = "hsa",
  species = "Homo sapiens",
  db_species = "HS",
  collection,
  subcollection = NULL
)
```

Arguments

`source`	As of this version, either 'KEGG', 'Reactome' or 'MSigDB' (default = 'KEGG')
`org_code`	(Used for 'KEGG' only) KEGG organism code for the selected organism. For a full list of all available organisms, see https://www.genome.jp/kegg/catalog/org_list.html
`species`	species name for output genes, such as Homo sapiens, Mus musculus, etc. See `msigdbr_species` for all the species available in the msigdbr package.
`db_species`	Species abbreviation for the human or mouse databases ("HS" or "MM").
`collection`	collection. e.g., H, C1. (default = NULL, i.e. list all gene sets in collection). See `msigdbr_collections` for all available options the msigdbr package.
`subcollection`	sub-collection, such as CGP, BP, etc. (default = NULL, i.e. list all gene sets in collection). See `msigdbr_collections` for all available options the msigdbr package.

Value

A list containing 2 elements:

- gene_sets - A list containing the genes involved in each gene set
- descriptions - A named vector containing the descriptions for each gene set

. For 'KEGG' and 'MSigDB', it is possible to choose a specific organism. For a full list of all available KEGG organisms, see https://www.genome.jp/kegg/catalog/org_list.html. See `msigdbr_species` for all the species available in the msigdbr package used for obtaining 'MSigDB' gene sets. For Reactome, there is only one collection of pathway gene sets.

get_kegg_gsets *Retrieve Organism-specific KEGG Pathway Gene Sets*

Description

Retrieve Organism-specific KEGG Pathway Gene Sets

Usage

```
get_kegg_gsets(org_code = "hsa")
```

Arguments

org_code	KEGG organism code for the selected organism. For a full list of all available organisms, see https://www.genome.jp/kegg/catalog/org_list.html

Value

list containing 2 elements:

- gene_sets - A list containing KEGG IDs for the genes involved in each KEGG pathway
- descriptions - A named vector containing the descriptions for each KEGG pathway

get_mgsigdb_gsets *Retrieve Organism-specific MSigDB Gene Sets*

Description

Retrieve Organism-specific MSigDB Gene Sets

Usage

```
get_mgsigdb_gsets(
  species = "Homo sapiens",
  db_species = "HS",
  collection = NULL,
  subcollection = NULL
)
```

Arguments

species	species name for output genes, such as Homo sapiens, Mus musculus, etc. See msigdbr_species for all the species available in the msigdbr package.
db_species	Species abbreviation for the human or mouse databases ("HS" or "MM").
collection	collection. e.g., H, C1. (default = NULL, i.e. list all gene sets in collection). See msigdbr_collections for all available options the msigdbr package.
subcollection	sub-collection, such as CGP, BP, etc. (default = NULL, i.e. list all gene sets in collection). See msigdbr_collections for all available options the msigdbr package.

Details

this function utilizes the function msigdbr from the msigdbr package to retrieve the 'Molecular Signatures Database' (MSigDB) gene sets (Subramanian et al. 2005 <doi:10.1073/pnas.0506580102>, Liberzon et al. 2015 <doi:10.1016/j.cels.2015.12.004>). Available collections are: H: hallmark gene sets, C1: positional gene sets, C2: curated gene sets, C3: motif gene sets, C4: computational gene sets, C5: GO gene sets, C6: oncogenic signatures and C7: immunologic signatures

Value

Retrieves the MSigDB gene sets and returns a list containing 2 elements:

- gene_sets - A list containing the genes involved in each of the selected MSigDB gene sets
- descriptions - A named vector containing the descriptions for each selected MSigDB gene set

---

get_pin_file *Retrieve Organism-specific PIN data*

Description

Retrieve Organism-specific PIN data

Usage

```
get_pin_file(source = "BioGRID", org = "Homo_sapiens", path2pin, ...)
```

Arguments

source	As of this version, this function is implemented to get data from 'BioGRID' only. This argument (and this wrapper function) was implemented for future utility
org	organism name. BioGRID naming requires underscores for spaces so 'Homo sapiens' becomes 'Homo_sapiens', 'Mus musculus' becomes 'Mus_musculus' etc. See https://wiki.thebiogrid.org/doku.php/statistics for a full list of available organisms (default = 'Homo_sapiens')
path2pin	the path of the file to save the PIN data. By default, the PIN data is saved in a temporary file
...	additional arguments for get_biogrid_pin

Value

the path of the file in which the PIN data was saved. If `path2pin` was not supplied by the user, the PIN data is saved in a temporary file

Examples

```
## Not run:
pin_path <- get_pin_file()

## End(Not run)
```

---

get_reactome_gsets *Retrieve Reactome Pathway Gene Sets*

Description

Retrieve Reactome Pathway Gene Sets

Usage

```
get_reactome_gsets()
```

Value

Gets the latest Reactome pathways gene sets in gmt format. Parses the gmt file and returns a list containing 2 elements:

- gene_sets - A list containing the genes involved in each Reactome pathway
- descriptions - A named vector containing the descriptions for each Reactome pathway

---

gset_list_from_gmt *Retrieve Gene Sets from GMT-format File*

Description

Retrieve Gene Sets from GMT-format File

Usage

```
gset_list_from_gmt(path2gmt, descriptions_idx = 2)
```

Arguments

`path2gmt`	path to the gmt file
`descriptions_idx`	index for descriptions (default = 2)

Value

list containing 2 elements:

- gene_sets - A list containing the genes involved in each gene set
- descriptions - A named vector containing the descriptions for each gene set

---

`hierarchical_term_clustering`
Hierarchical Clustering of Enriched Terms

Description

Hierarchical Clustering of Enriched Terms

Usage

```
hierarchical_term_clustering(
  kappa_mat,
  enrichment_res,
  num_clusters = NULL,
  use_description = FALSE,
  clu_method = "average",
  plot_hmap = FALSE,
  plot_dend = TRUE
)
```

Arguments

`kappa_mat`	matrix of kappa statistics (output of `create_kappa_matrix`)
`enrichment_res`	data frame of pathfindR enrichment results. Must-have columns are 'Term_Description' (if `use_description = TRUE`) or 'ID' (if `use_description = FALSE`), 'Down_regulated', and 'Up_regulated'. If `use_active_snw_genes = TRUE`, 'non_Signif_Snw_Genes' must also be provided.
`num_clusters`	number of clusters to be formed (default = `NULL`). If `NULL`, the optimal number of clusters is determined as the number which yields the highest average silhouette width.
`use_description`	Boolean argument to indicate whether term descriptions (in the 'Term_Description' column) should be used. (default = `FALSE`)
`clu_method`	the agglomeration method to be used (default = 'average', see `hclust`)
`plot_hmap`	boolean to indicate whether to plot the kappa statistics clustering heatmap or not (default = FALSE)
`plot_dend`	boolean to indicate whether to plot the clustering dendrogram partitioned into the optimal number of clusters (default = TRUE)

Details

The function initially performs hierarchical clustering of the enriched terms in `enrichment_res` using the kappa statistics (defining the distance as `1 - kappa_statistic`). Next, the clustering dendrogram is cut into k = 2, 3, ..., n - 1 clusters (where n is the number of terms). The optimal number of clusters is determined as the k value which yields the highest average silhouette width. (if `num_clusters` not specified)

Value

a vector of clusters for each enriched term in the enrichment results.

Examples

```
## Not run:
hierarchical_term_clustering(kappa_mat, enrichment_res)
hierarchical_term_clustering(kappa_mat, enrichment_res, method = "complete")

## End(Not run)
```

---

hyperg_test *Hypergeometric Distribution-based Hypothesis Testing*

Description

Hypergeometric Distribution-based Hypothesis Testing

Usage

```
hyperg_test(term_genes, chosen_genes, background_genes)
```

Arguments

`term_genes`	vector of genes in the selected term gene set
`chosen_genes`	vector containing the set of input genes
`background_genes`	vector of background genes (i.e. universal set of genes in the experiment)

Details

To determine whether the `chosen_genes` are enriched (compared to a background pool of genes) in the `term_genes`, the hypergeometric distribution is assumed and the appropriate p value (the value under the right tail) is calculated and returned.

Value

the p-value as determined using the hypergeometric distribution.

Examples

```
hyperg_test(letters[1:5], letters[2:5], letters)
hyperg_test(letters[1:5], letters[2:10], letters)
hyperg_test(letters[1:5], letters[2:13], letters)
```

`input_processing` *Process Input*

Description

Process Input

Usage

```
input_processing(
  input,
  p_val_threshold = 0.05,
  pin_name_path = "Biogrid",
  convert2alias = TRUE
)
```

Arguments

`input`	the input data that pathfindR uses. The input must be a data frame with three columns: 1. Gene Symbol (Gene Symbol) 2. Change value, e.g. log(fold change) (OPTIONAL) 3. p value, e.g. adjusted p value associated with differential expression
`p_val_threshold`	the p value threshold to use when filtering the input data frame. Must a numeric value between 0 and 1. (default = 0.05)
`pin_name_path`	Name of the chosen PIN or absolute/path/to/PIN.sif. If PIN name, must be one of c('Biogrid', 'STRING', 'GeneMania', 'IntAct', 'KEGG', 'mmu_STRING'). If path/to/PIN.sif, the file must comply with the PIN specifications. (Default = 'Biogrid')
`convert2alias`	boolean to indicate whether or not to convert gene symbols in the input that are not found in the PIN to an alias symbol found in the PIN (default = TRUE) IMPORTANT NOTE: the conversion uses human gene symbols/alias symbols.

Value

This function first filters the input so that all p values are less than or equal to the threshold. Next, gene symbols that are not found in the PIN are identified. If aliases of these gene symbols are found in the PIN, the symbols are converted to the corresponding aliases. The resulting data frame containing the original gene symbols, the updated symbols, change values and p values is then returned.

See Also

See run_pathfindR for the wrapper function of the pathfindR workflow

Examples

```
processed_df <- input_processing(
  input = example_pathfindR_input[1:5, ],
  pin_name_path = "KEGG"
)
processed_df <- input_processing(
  input = example_pathfindR_input[1:5, ],
  pin_name_path = "KEGG",
  convert2alias = FALSE
)
```

---

input_testing *Input Testing*

Description

Input Testing

Usage

```
input_testing(input, p_val_threshold = 0.05)
```

Arguments

input	the input data that pathfindR uses. The input must be a data frame with three columns: 1. Gene Symbol (Gene Symbol) 2. Change value, e.g. log(fold change) (OPTIONAL) 3. p value, e.g. adjusted p value associated with differential expression
p_val_threshold	the p value threshold to use when filtering the input data frame. Must a numeric value between 0 and 1. (default = 0.05)

Value

Only checks if the input and the threshold follows the required specifications.

See Also

See run_pathfindR for the wrapper function of the pathfindR workflow

Examples

```
input_testing(example_pathfindR_input, 0.05)
```

isColor *Check if value is a valid color*

Description

Check if value is a valid color

Usage

```
isColor(x)
```

Arguments

x	value

Value

TRUE if x is a valid color, otherwise FALSE

order_df_by_columnn *Order input data frame by provided columnn*

Description

Order input data frame by provided columnn

Usage

```
order_df_by_columnn(df, order_by)
```

Arguments

df	the input data frame to be ordered
order_by	A column name

Value

The ordered data frame or raises error

pathfindR *pathfindR: A package for Enrichment Analysis Utilizing Active Subnetworks*

Description

pathfindR is a tool for active-subnetwork-oriented gene set enrichment analysis. The main aim of the package is to identify active subnetworks in a protein-protein interaction network using a user-provided list of genes and associated p values then performing enrichment analyses on the identified subnetworks, discovering enriched terms (i.e. pathways, gene ontology, TF target gene sets etc.) that possibly underlie the phenotype of interest.

Details

For analysis on non-Homo sapiens organisms, pathfindR offers utility functions for obtaining organism-specific PIN data and organism-specific gene sets data.

pathfindR also offers functionalities to cluster the enriched terms and identify representative terms in each cluster, to score the enriched terms per sample and to visualize analysis results.

Author(s)

Maintainer: Ege Ulgen <egeulgen@gmail.com> (ORCID) [copyright holder]

Authors:

- Ozan Ozisik <ozanytu@gmail.com> (ORCID)

See Also

See `run_pathfindR` for details on the pathfindR active-subnetwork-oriented enrichment analysis See `cluster_enriched_terms` for details on methods of enriched terms clustering to define clusters of biologically-related terms See `score_terms` for details on agglomerated score calculation for enriched terms to investigate how a gene set is altered in a given sample (or in cases vs. controls) See `term_gene_heatmap` for details on visualization of the heatmap of enriched terms by involved genes See `create_term_gene_graph` and `create_term_gene_plot` for details on visualizing terms and term-related genes as a graph to determine the degree of overlap between the enriched terms by identifying shared and/or distinct significant genes See `UpSet_plot` for details on creating an UpSet plot of the enriched terms. See `get_pin_file` for obtaining organism-specific PIN data and `get_gene_sets_list` for obtaining organism-specific gene sets data

plot_scores *Plot the Heatmap of Score Matrix of Enriched Terms per Sample*

Description

Plot the Heatmap of Score Matrix of Enriched Terms per Sample

Usage

```
plot_scores(
  score_matrix,
  cases = NULL,
  label_samples = TRUE,
  case_title = "Case",
  control_title = "Control",
  low = "green",
  mid = "black",
  high = "red"
)
```

Arguments

score_matrix	Matrix of agglomerated enriched term scores per sample. Columns are samples, rows are enriched terms
cases	(Optional) A vector of sample names that are cases in the case/control experiment. (default = NULL)
label_samples	Boolean value to indicate whether or not to label the samples in the heatmap plot (default = TRUE)
case_title	Naming of the 'Case' group (as in cases) (default = 'Case')
control_title	Naming of the 'Control' group (default = 'Control')
low	a string indicating the color of 'low' values in the coloring gradient (default = 'green')
mid	a string indicating the color of 'mid' values in the coloring gradient (default = 'black')
high	a string indicating the color of 'high' values in the coloring gradient (default = 'red')

Value

A 'ggplot2' object containing the heatmap plot. x-axis indicates the samples. y-axis indicates the enriched terms. 'Score' indicates the score of the term in a given sample. If cases are provided, the plot is divided into 2 facets, named by case_title and control_title.

Examples

```
score_matrix <- score_terms(
  example_pathfindR_output,
  example_experiment_matrix,
  plot_hmap = FALSE
)
hmap <- plot_scores(score_matrix)
```

`process_pin` *Process Data frame of Protein-protein Interactions*

Description

Process Data frame of Protein-protein Interactions

Usage

```
process_pin(pin_df)
```

Arguments

`pin_df`	data frame of protein-protein interactions with 2 columns: 'Interactor_A' and 'Interactor_B'

Value

processed PIN data frame (removes self-interactions and duplicated interactions)

`return_pin_path` *Return The Path to Given Protein-Protein Interaction Network (PIN)*

Description

This function returns the absolute path/to/PIN.sif. While the default PINs are 'Biogrid', 'STRING', 'GeneMania', 'IntAct', 'KEGG' and 'mmu_STRING'. The user can also use any other PIN by specifying the 'path/to/PIN.sif'. All PINs to be used in this package must formatted as SIF files: i.e. have 3 columns with no header, no row names and be tab-separated. Columns 1 and 3 must be interactors' gene symbols, column 2 must be a column with all rows consisting of 'pp'.

Usage

```
return_pin_path(pin_name_path = "Biogrid")
```

Arguments

pin_name_path	Name of the chosen PIN or absolute/path/to/PIN.sif. If PIN name, must be one of c('Biogrid', 'STRING', 'GeneMania', 'IntAct', 'KEGG', 'mmu_STRING'). If path/to/PIN.sif, the file must comply with the PIN specifications. (Default = 'Biogrid')

Value

The absolute path to chosen PIN.

See Also

See run_pathfindR for the wrapper function of the pathfindR workflow

Examples

```
## Not run:
pin_path <- return_pin_path("GeneMania")

## End(Not run)
```

---

run_pathfindR *Wrapper Function for pathfindR - Active-Subnetwork-Oriented Enrichment Workflow*

Description

run_pathfindR is the wrapper function for the pathfindR workflow

Usage

```
run_pathfindR(
  input,
  gene_sets = "KEGG",
  min_gset_size = 10,
  max_gset_size = 300,
  custom_genes = NULL,
  custom_descriptions = NULL,
  pin_name_path = "Biogrid",
  p_val_threshold = 0.05,
  enrichment_threshold = 0.05,
  convert2alias = TRUE,
  plot_enrichment_chart = TRUE,
  output_dir = NULL,
  list_active_snw_genes = FALSE,
  ...
)
```

Arguments

input	the input data that pathfindR uses. The input must be a data frame with three columns: 1. Gene Symbol (Gene Symbol) 2. Change value, e.g. log(fold change) (OPTIONAL) 3. p value, e.g. adjusted p value associated with differential expression
gene_sets	Name of the gene sets to be used for enrichment analysis. Available gene sets are 'KEGG', 'Reactome', 'BioCarta', 'GO-All', 'GO-BP', 'GO-CC', 'GO-MF', 'cell_markers', 'mmu_KEGG' or 'Custom'. If 'Custom', the arguments custom_genes and custom_descriptions must be specified. (Default = 'KEGG')
min_gset_size	minimum number of genes a term must contain (default = 10)
max_gset_size	maximum number of genes a term must contain (default = 300)
custom_genes	a list containing the genes involved in each custom term. Each element is a vector of gene symbols located in the given custom term. Names should correspond to the IDs of the custom terms.
custom_descriptions	A vector containing the descriptions for each custom term. Names of the vector should correspond to the IDs of the custom terms.
pin_name_path	Name of the chosen PIN or absolute/path/to/PIN.sif. If PIN name, must be one of c('Biogrid', 'STRING', 'GeneMania', 'IntAct', 'KEGG', 'mmu_STRING'). If path/to/PIN.sif, the file must comply with the PIN specifications. (Default = 'Biogrid')
p_val_threshold	the p value threshold to use when filtering the input data frame. Must a numeric value between 0 and 1. (default = 0.05)
enrichment_threshold	adjusted-p value threshold used when filtering enrichment results (default = 0.05)
convert2alias	boolean to indicate whether or not to convert gene symbols in the input that are not found in the PIN to an alias symbol found in the PIN (default = TRUE) IMPORTANT NOTE: the conversion uses human gene symbols/alias symbols.
plot_enrichment_chart	boolean value. If TRUE, a bubble chart displaying the enrichment results is plotted. (default = TRUE)
output_dir	the directory to be created where the output and intermediate files are saved (default = NULL, a temporary directory is used)
list_active_snw_genes	boolean value indicating whether or not to report the non-significant active subnetwork genes for the active subnetwork which was enriched for the given term with the lowest p value (default = FALSE)
...	additional arguments for active_snw_enrichment_wrapper

Details

This function takes in a data frame consisting of Gene Symbol, log-fold-change and adjusted-p values. After input testing, any gene symbols that are not in the PIN are converted to alias symbols if the alias is in the PIN. Next, active subnetwork search is performed. Enrichment analysis is performed using the genes in each of the active subnetworks. Terms with adjusted-p values lower than `enrichment_threshold` are discarded. The lowest adjusted-p value (over all subnetworks) for each term is kept. This process of active subnetwork search and enrichment is repeated for a selected number of `iterations`, which is done in parallel. Over all iterations, the lowest and the highest adjusted-p values, as well as number of occurrences are reported for each enriched term.

Value

Data frame of pathfindR enrichment results. Columns are:

ID ID of the enriched term

Term_Description Description of the enriched term

Fold_Enrichment Fold enrichment value for the enriched term (Calculated using ONLY the input genes)

occurrence the number of iterations that the given term was found to enriched over all iterations

support the median support (proportion of active subnetworks leading to enrichment within an iteration) over all iterations

lowest_p the lowest adjusted-p value of the given term over all iterations

highest_p the highest adjusted-p value of the given term over all iterations

non_Signif_Snw_Genes (OPTIONAL) the non-significant active subnetwork genes, comma-separated

Up_regulated the up-regulated genes (as determined by 'change value' > 0, if the 'change column' was provided) in the input involved in the given term's gene set, comma-separated. If change column not provided, all affected are listed here.

Down_regulated the down-regulated genes (as determined by 'change value' < 0, if the 'change column' was provided) in the input involved in the given term's gene set, comma-separated

The function also creates an HTML report with the pathfindR enrichment results linked to the visualizations of the enriched terms in addition to the table of converted gene symbols. This report can be found in 'output_dir/results.html' under the current working directory.

By default, a bubble chart of top 10 enrichment results are plotted. The x-axis corresponds to fold enrichment values while the y-axis indicates the enriched terms. Sizes of the bubbles indicate the number of significant genes in the given terms. Color indicates the -log10(lowest-p) value; the more red it is, the more significant the enriched term is. See `enrichment_chart`.

Warning

Especially depending on the protein interaction network, the algorithm and the number of iterations you choose, 'active subnetwork search + enrichment' component of `run_pathfindR` may take a long time to finish.

See Also

input_testing for input testing, input_processing for input processing, get_active_subnetworks for active subnetwork search and subnetwork filtering, enrichment_analyses for enrichment analysis (using the active subnetworks), summarize_enrichment_results for summarizing the active-subnetwork-oriented enrichment results, annotate_term_genes for annotation of affected genes in the given gene sets, visualize_terms for visualization of enriched terms, enrichment_chart for a visual summary of the pathfindR enrichment results, foreach for details on parallel execution of looping constructs, cluster_enriched_terms for clustering the resulting enriched terms and partitioning into clusters.

Examples

```
## Not run:
run_pathfindR(example_pathfindR_input)

## End(Not run)
```

---

safe_get_content *Safely download and parse web content*

Description

This helper function retrieves content from a given URL using **httr**. It ensures that common issues (e.g. no internet, timeouts, HTTP errors, or parsing errors) are handled gracefully with clear, informative error messages.

Usage

```
safe_get_content(url, ..., timeout_sec = 10)
```

Arguments

url	Character string. The URL of the resource to download.
...	Additional arguments passed to GET.
timeout_sec	Numeric. Timeout in seconds for the request (default = 10).

Details

This function is intended for use inside package functions. For examples, vignettes, or tests, wrap calls in a connectivity check (e.g. using http_error(HEAD(url))) to avoid CRAN failures when the resource is temporarily unavailable.

Value

A character string containing the parsed content of the response (UTF-8 encoded). On failure, an error is raised with a clear message.

Examples

```
## Not run:
# Retrieve the latest BioGRID release page
result <- safe_get_content("https://downloads.thebiogrid.org/BioGRID/Latest-Release/")

## End(Not run)
```

`score_terms` *Calculate Agglomerated Scores of Enriched Terms for Each Subject*

Description

Calculate Agglomerated Scores of Enriched Terms for Each Subject

Usage

```
score_terms(
  enrichment_table,
  exp_mat,
  cases = NULL,
  use_description = FALSE,
  plot_hmap = TRUE,
  ...
)
```

Arguments

`enrichment_table`
: a data frame that must contain the 3 columns below:

  Term_Description Description of the enriched term (necessary if `use_description = TRUE`)

  ID ID of the enriched term (necessary if `use_description = FALSE`)

  Up_regulated the up-regulated genes in the input involved in the given term's gene set, comma-separated

  Down_regulated the down-regulated genes in the input involved in the given term's gene set, comma-separated

`exp_mat`
: the experiment (e.g., gene expression/methylation) matrix. Columns are samples and rows are genes. Column names must contain sample names and row names must contain the gene symbols.

`cases`
: (Optional) A vector of sample names that are cases in the case/control experiment. (default = NULL)

`use_description`
: Boolean argument to indicate whether term descriptions (in the 'Term_Description' column) should be used. (default = `FALSE`)

plot_hmap	Boolean value to indicate whether or not to draw the heatmap plot of the scores. (default = TRUE)
...	Additional arguments for plot_scores for aesthetics of the heatmap plot

Value

Matrix of agglomerated scores of each enriched term per sample. Columns are samples, rows are enriched terms. Optionally, displays a heatmap of this matrix.

Conceptual Background

For an experiment matrix (containing expression, methylation, etc. values), the rows of which are genes and the columns of which are samples, we denote:

- E as a matrix of size $m \times n$
- G as the set of all genes in the experiment $G = E_{i\cdot}, \ \ i \in [1, m]$
- S as the set of all samples in the experiment $S = E_{j\cdot}, \ \ \in [1, n]$

We next define the gene score matrix GS (the standardized experiment matrix, also of size $m \times n$) as:

$GS_{gs} = \frac{E_{gs} - \bar{e_g}}{s_g}$

where $g \in G$, $s \in S$, $\bar{e_g}$ is the mean of all values for gene g and $\bar{s_g}$ is the standard deviation of all values for gene g.

We next denote T to be a set of terms (where each $t \in T$ is a set of term-related genes, i.e., $t = \{g_x, ..., g_y\} \subset G$) and finally define the agglomerated term scores matrix TS (where rows correspond to genes and columns corresponds to samples s.t. the matrix has size $|T| \times n$) as:

$TS_{ts} = \frac{1}{|t|} \sum_{g \in t} GS_{gs}$, where $t \in T$ and $s \in S$.

Examples

```
score_matrix <- score_terms(
  example_pathfindR_output,
  example_experiment_matrix,
  plot_hmap = FALSE
)
```

---

single_iter_wrapper *Active Subnetwork Search + Enrichment Analysis Wrapper for a Single Iteration*

Description

Active Subnetwork Search + Enrichment Analysis Wrapper for a Single Iteration

Usage

```
single_iter_wrapper(
  i = NULL,
  pin_path,
  network,
  experiment_df,
  score_quan_thr,
  sig_gene_thr,
  search_method,
  verbose,
  start_with_all_positives,
  gene_init_prob,
  sa_initial_temp,
  sa_final_temp,
  sa_iterations,
  ga_population_size,
  ga_iterations,
  ga_crossover_rate,
  ga_mutation_rate,
  gr_max_depth,
  gr_search_depth,
  gr_overlap_threshold,
  gr_subnetwork_num,
  gset_list,
  adj_method,
  enrichment_threshold,
  list_active_snw_genes
)
```

Arguments

i	current iteration index (default = NULL)
pin_path	path/to/PIN/file
network	Prebuilt network object as returned by build_network(), built once by active_snw_enrichment_wrapp and passed to every iteration to avoid redundant PIN file I/O.
experiment_df	input experiment data frame
score_quan_thr	active subnetwork score quantile threshold. Must be between 0 and 1 or set to -1 for not filtering. (Default = 0.8)
sig_gene_thr	threshold for the minimum proportion of significant genes in the subnetwork (Default = 0.02) If the number of genes to use as threshold is calculated to be < 2 (e.g. 50 signif. genes x 0.01 = 0.5), the threshold number is set to 2
search_method	algorithm to use when performing active subnetwork search. Options are greedy search (GR), simulated annealing (SA) or genetic algorithm (GA) for the search (default = 'GR').
verbose	boolean value indicating whether to print messages (default=FALSE)

`start_with_all_positives`
: if TRUE: in GA, adds an individual with all positive nodes. In SA, initializes candidate solution with all positive nodes. (default = FALSE)

`gene_init_prob` For SA and GA, probability of adding a gene in initial solution (default = 0.1)

`sa_initial_temp`
: Initial temperature for SA (default = 1.0)

`sa_final_temp` Final temperature for SA (default = 0.01)

`sa_iterations` Iteration number for SA (default = 10000)

`ga_population_size`
: Population size for GA (default = 400)

`ga_iterations` Iteration number for GA (default = 200)

`ga_crossover_rate`
: Applies crossover with the given probability in GA (default = 1, i.e. always perform crossover)

`ga_mutation_rate`
: For GA, applies mutation with given mutation rate (default = 0, i.e. mutation off)

`gr_max_depth` Sets max depth in greedy search, 0 for no limit (default = 1)

`gr_search_depth`
: Search depth in greedy search (default = 1)

`gr_overlap_threshold`
: Overlap threshold for results of greedy search (default = 0.5)

`gr_subnetwork_num`
: Number of subnetworks to be presented in the results (default = 1000)

`gset_list` list for gene sets.

`adj_method` correction method to be used for adjusting p-values. (default = 'bonferroni')

`enrichment_threshold`
: adjusted-p value threshold used when filtering enrichment results (default = 0.05)

`list_active_snw_genes`
: boolean value indicating whether or not to report the non-significant active subnetwork genes for the active subnetwork which was enriched for the given term with the lowest p value (default = `FALSE`)

Value

Data frame of enrichment results using active subnetwork search results

summarize_enrichment_results *Summarize Enrichment Results*

Description

Summarize Enrichment Results

Usage

```
summarize_enrichment_results(enrichment_res, list_active_snw_genes = FALSE)
```

Arguments

enrichment_res	a dataframe of combined enrichment results. Columns are: **ID** ID of the enriched term **Term_Description** Description of the enriched term **Fold_Enrichment** Fold enrichment value for the enriched term **p_value** p value of enrichment **adj_p** adjusted p value of enrichment **non_Signif_Snw_Genes (OPTIONAL)** the non-significant active subnetwork genes, comma-separated
list_active_snw_genes	boolean value indicating whether or not to report the non-significant active subnetwork genes for the active subnetwork which was enriched for the given term with the lowest p value (default = FALSE)

Value

a dataframe of summarized enrichment results (over multiple iterations). Columns are:

ID ID of the enriched term

Term_Description Description of the enriched term

Fold_Enrichment Fold enrichment value for the enriched term

occurrence the number of iterations that the given term was found to enriched over all iterations

support the median support (proportion of active subnetworks leading to enrichment within an iteration) over all iterations

lowest_p the lowest adjusted-p value of the given term over all iterations

highest_p the highest adjusted-p value of the given term over all iterations

non_Signif_Snw_Genes (OPTIONAL) the non-significant active subnetwork genes, comma-separated

Examples

```
## Not run:
summarize_enrichment_results(enrichment_res)

## End(Not run)
```

`term_gene_heatmap` *Create Terms by Genes Heatmap*

Description

Create Terms by Genes Heatmap

Usage

```
term_gene_heatmap(
  result_df,
  genes_df,
  num_terms = 10,
  use_description = FALSE,
  low = "red",
  mid = "black",
  high = "green",
  legend_title = "change",
  sort_terms_by_p = FALSE,
  ...
)
```

Arguments

`result_df`	A dataframe of pathfindR results that must contain the following columns: **Term_Description** Description of the enriched term (necessary if `use_description = TRUE`) **ID** ID of the enriched term (necessary if `use_description = FALSE`) **lowest_p** the highest adjusted-p value of the given term over all iterations **Up_regulated** the up-regulated genes in the input involved in the given term's gene set, comma-separated **Down_regulated** the down-regulated genes in the input involved in the given term's gene set, comma-separated
`genes_df`	the input data that was used with `run_pathfindR`. It must be a data frame with 3 columns: 1. Gene Symbol (Gene Symbol) 2. Change value, e.g. log(fold change) (optional) 3. p value, e.g. adjusted p value associated with differential expression The change values in this data frame are used to color the affected genes
`num_terms`	Number of top enriched terms to use while creating the plot. Set to `NULL` to use all enriched terms (default = 10)
`use_description`	Boolean argument to indicate whether term descriptions (in the 'Term_Description' column) should be used. (default: `FALSE`)

low	a string indicating the color of 'low' values in the coloring gradient (default = 'green')
mid	a string indicating the color of 'mid' values in the coloring gradient (default = 'black')
high	a string indicating the color of 'high' values in the coloring gradient (default = 'red')
legend_title	legend title (default = 'change')
sort_terms_by_p	boolean to indicate whether to sort terms by 'lowest_p' (TRUE) or by number of genes (FALSE) (default = FALSE)
...	additional arguments for input_processing (used if genes_df is provided)

Value

a ggplot2 object of a heatmap where rows are enriched terms and columns are involved input genes. If genes_df is provided, colors of the tiles indicate the change values.

Examples

```
term_gene_heatmap(example_pathfindR_output, num_terms = 3)
```

UpSet_plot *Create UpSet Plot of Enriched Terms*

Description

Create UpSet Plot of Enriched Terms

Usage

```
UpSet_plot(
  result_df,
  genes_df,
  num_terms = 10,
  method = "heatmap",
  use_description = FALSE,
  low = "red",
  mid = "black",
  high = "green",
  ...
)
```

Arguments

result_df	A dataframe of pathfindR results that must contain the following columns: **Term_Description** Description of the enriched term (necessary if use_description = TRUE) **ID** ID of the enriched term (necessary if use_description = FALSE) **lowest_p** the highest adjusted-p value of the given term over all iterations **Up_regulated** the up-regulated genes in the input involved in the given term's gene set, comma-separated **Down_regulated** the down-regulated genes in the input involved in the given term's gene set, comma-separated
genes_df	the input data that was used with run_pathfindR. It must be a data frame with 3 columns: 1. Gene Symbol (Gene Symbol) 2. Change value, e.g. log(fold change) (optional) 3. p value, e.g. adjusted p value associated with differential expression The change values in this data frame are used to color the affected genes
num_terms	Number of top enriched terms to use while creating the plot. Set to NULL to use all enriched terms (default = 10)
method	the option for producing the plot. Options include 'heatmap', 'boxplot' and 'barplot'. (default = 'heatmap')
use_description	Boolean argument to indicate whether term descriptions (in the 'Term_Description' column) should be used. (default: FALSE)
low	a string indicating the color of 'low' values in the coloring gradient (default = 'green')
mid	a string indicating the color of 'mid' values in the coloring gradient (default = 'black')
high	a string indicating the color of 'high' values in the coloring gradient (default = 'red')
...	additional arguments for input_processing (used if genes_df is provided)

Value

UpSet plots are plots of the intersections of sets as a matrix. This function creates a ggplot object of an UpSet plot where the x-axis is the UpSet plot of intersections of enriched terms. By default (i.e. method = 'heatmap') the main plot is a heatmap of genes at the corresponding intersections, colored by up/down regulation (if genes_df is provided, colored by change values). If method = 'barplot', the main plot is bar plots of the number of genes at the corresponding intersections. Finally, if method = 'boxplot' and if genes_df is provided, then the main plot displays the boxplots of change values of the genes at the corresponding intersections.

Examples

```
UpSet_plot(example_pathfindR_output)
```

`visualize_active_subnetworks` *Visualize Active Subnetworks*

Description

Visualize Active Subnetworks

Usage

```
visualize_active_subnetworks(
  active_snws,
  genes_df,
  pin_name_path = "Biogrid",
  num_snws,
  layout = "stress",
  score_quan_thr = 0.8,
  sig_gene_thr = 0.02,
  ...
)
```

Arguments

`active_snws`	active subnetwork search results. A list containing input `subnetworks` (nodes) and `scores` (score).
`genes_df`	the input data that was used with `run_pathfindR`. It must be a data frame with 3 columns: 1. Gene Symbol (Gene Symbol) 2. Change value, e.g. log(fold change) (optional) 3. p value, e.g. adjusted p value associated with differential expression The change values in this data frame are used to color the affected genes
`pin_name_path`	Name of the chosen PIN or absolute/path/to/PIN.sif. If PIN name, must be one of c('Biogrid', 'STRING', 'GeneMania', 'IntAct', 'KEGG', 'mmu_STRING'). If path/to/PIN.sif, the file must comply with the PIN specifications. (Default = 'Biogrid')
`num_snws`	number of top subnetworks to be visualized (leave blank if you want to visualize all subnetworks)
`layout`	The type of layout to create (see `ggraph` for details (default: `'stress'`)
`score_quan_thr`	active subnetwork score quantile threshold. Must be between 0 and 1 or set to -1 for not filtering. (Default = 0.8)
`sig_gene_thr`	threshold for the minimum proportion of significant genes in the subnetwork (Default = 0.02) If the number of genes to use as threshold is calculated to be < 2 (e.g. 50 signif. genes x 0.01 = 0.5), the threshold number is set to 2
`...`	additional arguments for `input_processing`

Value

a list of ggplot objects of graph visualizations of identified active subnetworks. Green nodes are down-regulated genes, reds are up-regulated genes and yellows are non-input genes

Examples

```
# visualize top 2 active subnetworks
g_list <- visualize_active_subnetworks(
  active_snws = example_unfiltered_snws,
  genes_df = example_pathfindR_input[1:10, ],
  pin_name_path = "KEGG",
  num_snws = 2
)
```

`visualize_KEGG_diagram` *Visualize Human KEGG Pathways*

Description

Visualize Human KEGG Pathways

Usage

```
visualize_KEGG_diagram(
  kegg_pw_ids,
  input_processed,
  scale_vals = TRUE,
  node_cols = NULL,
  legend.position = "top"
)
```

Arguments

`kegg_pw_ids`	KEGG ids of pathways to be colored and visualized
`input_processed`	input data processed via `input_processing`
`scale_vals`	should change values be scaled? (default = `TRUE`)
`node_cols`	low, middle and high color values for coloring the pathway nodes (default = `NULL`). If `node_cols=NULL`, the low, middle and high color are set as 'green', 'gray' and 'red'. If all change values are 1e6 (in case no changes are supplied, this dummy value is assigned by `input_processing`), only one color ('#F38F18' if NULL) is used.
`legend.position`	the default position of legends ("none", "left", "right", "bottom", "top", "inside")

Value

Creates colored visualizations of the enriched human KEGG pathways and returns them as a list of ggplot objects, named by Term ID.

See Also

See `visualize_terms` for the wrapper function for creating enriched term diagrams. See `run_pathfindR` for the wrapper function of the pathfindR enrichment workflow.

Examples

```
## Not run:
input_processed <- data.frame(
  GENE = c("PKLR", "GPI", "CREB1", "INS"),
  CHANGE = c(1.5, -2, 3, 5)
)
gg_list <- visualize_KEGG_diagram(c("hsa00010", "hsa04911"), input_processed)

## End(Not run)
```

---

visualize_term_interactions

Visualize Interactions of Genes Involved in the Given Enriched Terms

---

Description

Visualize Interactions of Genes Involved in the Given Enriched Terms

Usage

```
visualize_term_interactions(result_df, pin_name_path, show_legend = TRUE)
```

Arguments

`result_df`	Data frame of enrichment results. Must-have columns are: 'Term_Description', 'Up_regulated' and 'Down_regulated'
`pin_name_path`	Name of the chosen PIN or absolute/path/to/PIN.sif. If PIN name, must be one of c('Biogrid', 'STRING', 'GeneMania', 'IntAct', 'KEGG', 'mmu_STRING'). If path/to/PIN.sif, the file must comply with the PIN specifications. (Default = 'Biogrid')
`show_legend`	Boolean to indicate whether to display the legend (`TRUE`) or not (`FALSE`) (default: `TRUE`)

Details

The following steps are performed for the visualization of interactions of genes involved for each enriched term:

1. shortest paths between all affected genes are determined (via igraph)
2. the nodes of all shortest paths are merged
3. the PIN is subsetted using the merged nodes (genes)
4. using the PIN subset, the graph showing the interactions is generated
5. the final graph is visualized using igraph, colored by changed status (if provided)

Value

list of ggplot objects (named by Term ID) visualizing the interactions of genes involved in the given enriched terms (annotated in the `result_df`) in the PIN used for enrichment analysis (specified by `pin_name_path`).

See Also

See `visualize_terms` for the wrapper function for creating enriched term diagrams. See `run_pathfindR` for the wrapper function of the pathfindR enrichment workflow.

Examples

```
## Not run:
result_df <- example_pathfindR_output[1:2, ]
gg_list <- visualize_term_interactions(result_df, pin_name_path = "IntAct")

## End(Not run)
```

---

visualize_terms *Create Diagrams for Enriched Terms*

Description

Create Diagrams for Enriched Terms

Usage

```
visualize_terms(
  result_df,
  input_processed = NULL,
  is_KEGG_result = TRUE,
  pin_name_path = "Biogrid",
  ...
)
```

Arguments

result_df	Data frame of enrichment results. Must-have columns for KEGG human pathway diagrams (is_KEGG_result = TRUE) are: 'ID' and 'Term_Description'. Must-have columns for the rest are: 'Term_Description', 'Up_regulated' and 'Down_regulated'
input_processed	input data processed via input_processing, not necessary when is_KEGG_result = FALSE
is_KEGG_result	boolean to indicate whether KEGG gene sets were used for enrichment analysis or not (default = TRUE)
pin_name_path	Name of the chosen PIN or absolute/path/to/PIN.sif. If PIN name, must be one of c('Biogrid', 'STRING', 'GeneMania', 'IntAct', 'KEGG', 'mmu_STRING'). If path/to/PIN.sif, the file must comply with the PIN specifications. (Default = 'Biogrid')
...	additional arguments for visualize_KEGG_diagram (used when is_KEGG_result = TRUE) or visualize_term_interactions (used when is_KEGG_result = FALSE)

Details

For is_KEGG_result = TRUE, KEGG pathway diagrams are created, affected nodes colored by up/down regulation status. For other gene sets, interactions of affected genes are determined (via a shortest-path algorithm) and are visualized (colored by change status) using igraph.

Value

Depending on the argument is_KEGG_result, creates visualization of interactions of genes involved in the list of enriched terms in result_df. Returns a list of ggplot objects named by Term ID.

See Also

See visualize_KEGG_diagram for the visualization function of KEGG diagrams. See visualize_term_interactions for the visualization function that generates diagrams showing the interactions of input genes in the PIN. See run_pathfindR for the wrapper function of the pathfindR workflow.

Examples

```
## Not run:
input_processed <- data.frame(
  GENE = c("PARP1", "NDUFA1", "STX6", "SNAP23"),
  CHANGE = c(1.5, -2, 3, 5)
)
result_df <- example_pathfindR_output[1:2, ]

gg_list <- visualize_terms(result_df, input_processed)
gg_list2 <- visualize_terms(result_df, is_KEGG_result = FALSE, pin_name_path = "IntAct")

## End(Not run)
```

Index