

Package: genewalkR (via r-universe)

October 7, 2026

Title Graph-based computational algorithms in R, including diffusions and small neural networks like node2vec.

Version 0.2.1

Description Various graph-based methods commonly used in computational biology. Heat diffusions over networks, node2vec and metapath2vec-based analyses over biologicals networks in R powered by Rust. Includes GeneWalkR and co-expression gene drift detection methods based on node2vec.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/rextendr/version 0.4.2

SystemRequirements Cargo (Rust's package manager), rustc

Imports checkmate, duckdb, DBI, data.table, igraph, ggplot2, gridExtra, plotly, purrr, magrittr, rextendr, R6, S7, stringr, zeallot

Depends R (>= 4.2)

Suggests here, knitr, quarto, msigdb, tinytest

VignetteBuilder quarto

LazyData true

URL <https://gregorlueg.github.io/genewalkR/>

Config/roxygen2/version 8.1.0

Config/pak/sysreqs
cmake libglib-dev make libicu-dev libuv1-dev libxml2-dev libssl-dev xz-utils libclang-dev

Repository <https://gregorlueg.r-universe.dev>

Date/Publication 2026-10-07 06:01:25 UTC

RemoteUrl <https://github.com/GregorLueg/genewalkR>

RemoteRef main

RemoteSha 056a20e8c16e63f5794f1cccd5816e549f3620fa

Contents

build_kernel	3
calculate_drift	4
calculate_genewalk_stats	5
calculate_profile_distances	6
check_degree_distribution	7
dedup_edges	7
differential_graph_test_data	8
diffuse_scores	9
DiffusionProfiles	10
DiffusionScores	11
download_database	11
EmbedDrift	12
evaluate_node2vec_test	13
generate_initial_emb	13
generate_initial_embeddings	14
generate_permuted_emb	15
generate_profiles	16
GeneWalk	17
GeneWalkGenerator	18
get_aligned_embd	20
get_db_connection	20
get_embedding	21
get_embeddings	21
get_gene_info	22
get_gene_ontology_hierarchy	22
get_gene_ontology_info	23
get_gene_pathway_dt	23
get_gene_to_go	24
get_gene_to_reactome	24
get_graph_dt	25
get_graph_dts	25
get_gw_data	26
get_gw_data_filtered	26
get_interactions_combined	27
get_interactions_intact	28
get_interactions_pc	28
get_interactions_reactome	29
get_interactions_signor	29
get_interactions_string	30
get_profiles	30
get_reactome_hierarchy	31
get_reactome_info	31
get_scores	32
get_stats	32
metapath2vec	33
myc_genes	34

n_nodes	35
new_data_builder	35
node2vec	36
node2vec_test_data	37
params_diffusion_profiles	38
params_genewalk	38
params_genewalk_data	40
params_kernel	41
params_metapath2vec	42
params_node2vec	43
plot_gw_results	44
plot_similarities	45
reload_db	45
rs_build_synthetic_genewalk	45
rs_cosine_sim	46
rs_differential_graph_data	47
rs_diffuse	48
rs_diffusion_kernel	49
rs_diffusion_profiles	50
rs_gene_walk	51
rs_gene_walk_perm	52
rs_gene_walk_test	53
rs_kernel_node_names	53
rs_metapath2vec	54
rs_node2vec	55
rs_node2vec_synthetic_data	56
rs_procrustes_align	57
rs_profile_distances	57
synthetic_genewalk_data	58

Index 59

build_kernel	<i>Build the diffusion kernel</i>
--------------	-----------------------------------

Description

Constructs the graph Laplacian and runs the spectral decomposition, then applies the chosen kernel transform. The result is stored as an external pointer for subsequent calls to `diffuse()`.

Usage

```
build_kernel(
  object,
  kernel = c("regularised_laplacian", "commute_time", "diffusion", "inverse_cosine",
    "pstep"),
  kernel_params = params_kernel(),
  normalised = TRUE,
```

```

    strategy = "truncated",
    k = 100L,
    .verbose = TRUE
  )

```

Arguments

object	A DiffusionScores object, see DiffusionScores() .
kernel	Character. One of "regularised_laplacian", "commute_time", "diffusion", "inverse_cosine", "pstep". Defaults to "regularised_laplacian".
kernel_params	Named list. Kernel-specific parameters, see params_kernel() .
normalised	Logical. Use the normalised Laplacian. Defaults to TRUE.
strategy	Character. "full" or "truncated". Defaults to "truncated".
k	Integer. Number of eigenvalues for the truncated strategy. Ignored if strategy = "full". Defaults to 100L.
.verbose	Logical. Controls verbosity. Defaults to TRUE.

Value

The DiffusionScores object with kernel_pointer and params populated.

calculate_drift	<i>Calculate embedding drift between two graphs</i>
-----------------	---

Description

Aligns the embedding of graph 2 onto the reference embedding of graph 1 via orthogonal Procrustes, then computes per-node cosine similarities as a measure of context drift. Nodes shared between both graphs receive a cosine similarity in $[-1, 1]$. Nodes exclusive to graph 1 are assigned -1.1 and nodes exclusive to graph 2 are assigned $+1.1$ as sentinel values indicating absence rather than a computed similarity.

Usage

```
calculate_drift(object)
```

Arguments

object	An EmbedDrift object, see EmbedDrift() .
--------	--

Value

The EmbedDrift object with aligned_embd and stats populated. stats is a data.table with columns node, cosine_similarity, and node_status ("shared", "g1_only", "g2_only").

`calculate_genewalk_stats`*Calculate the pathway and gene statistics*

Description

Calculates statistical significance for gene-pathway associations using the GeneWalk algorithm. For each gene-pathway pair, this function computes p-values and FDR corrections (both globally across all pairs and per-gene) across multiple network permutations. The final output includes mean values and 95% confidence intervals for all statistics.

The function processes all permutations in Rust for optimal performance, calculating cosine similarities between gene and pathway embeddings, determining p-values based on null distributions from randomised networks, and applying Benjamini-Hochberg FDR corrections.

Usage

```
calculate_genewalk_stats(object, .verbose = TRUE)
```

Arguments

<code>object</code>	The GeneWalk object containing actual and permuted embeddings. See GeneWalk() .
<code>.verbose</code>	Logical. If TRUE, prints progress messages during calculation. Default is TRUE.

Value

The input object with an added `stats` property containing a `data.table` with the following columns for each gene-pathway pair:

- `gene` - Gene identifier
- `pathway` - Pathway identifier
- `avg_pval` - Mean p-value across permutations
- `pval_ci_lower`, `pval_ci_upper` - 95% CI for p-values
- `avg_global_fdr` - Mean global FDR across permutations
- `global_fdr_ci_lower`, `global_fdr_ci_upper` - 95% CI for global FDR
- `avg_gene_fdr` - Mean gene-wise FDR across permutations
- `gene_fdr_ci_lower`, `gene_fdr_ci_upper` - 95% CI for gene-wise FDR

References

Ietswaart, et al., Genome Biol, 2021

 calculate_profile_distances

Calculate distances between diffusion profiles

Description

Distances between two sets of stored profiles, e.g. drugs against diseases. Lower means more similar. Ruiz et al. rank drug-disease pairs by the correlation distance on the multiscale interactome and by Canberra when comparing against gene expression signatures.

Usage

```
calculate_profile_distances(
  object,
  from,
  to,
  metric = c("correlation", "canberra", "l1", "l2", "cosine")
)
```

Arguments

object	A DiffusionProfiles object with generated profiles.
from	Character vector. Seeds for the rows of the result.
to	Character vector. Seeds for the columns of the result.
metric	Character. One of "correlation", "canberra", "l1", "l2" or "cosine". Defaults to "correlation".

Details

Every profile sums to 1 and the seed keeps roughly $1 - \alpha$ of the mass on itself. What each metric weights:

- "correlation": Pearson on the full vector. Driven by the large entries, i.e. the immediate neighbourhood of the seed and hubs.
- "cosine": Without centring. Since all profiles sum to 1 and are mostly near zero, it ranks almost identically to "correlation".
- "l1": Twice the total variation distance between the two distributions. Less dominated by the top entries than the above.
- "l2": Dominated by the largest entries; favours big, well-connected nodes.
- "canberra": Relative differences per node, so the many near-zero entries count as much as the large ones. Useful when the profile is compared against a signature on a different scale.

Value

A numeric matrix of $\text{length}(\text{from}) \times \text{length}(\text{to})$ distances.

References

Ruiz, Zitnik and Leskovec, Identification of disease treatment mechanisms through the multiscale interactome, Nat Commun 2021.

check_degree_distribution

Check the degree across different node types

Description

Shows the degrees of the different edge types in the GeneWalk network. Lack of PPI interactions for example will make it difficult for the method to identify patterns.

Usage

```
check_degree_distribution(object)
```

Arguments

object The GeneWalk class, please see [GeneWalk\(\)](#).

dedup_edges

Deduplicate edges

Description

Collapses duplicate edges into one. The constrained PageRank keeps parallel edges, so a pair listed twice (e.g. the same gene pair from two interaction sources, or A-B plus B-A in an undirected graph) gets twice the transition weight. Run this before [DiffusionProfiles\(\)](#) unless that is what you want.

Usage

```
dedup_edges(
  graph_dt,
  node_dt = NULL,
  directed = FALSE,
  weight_agg = c("max", "sum", "mean"),
  .verbose = TRUE
)
```

Arguments

graph_dt	data.table. The edge table with "from", "to" and optionally "weight" (non-negative).
node_dt	Optional data.table with the columns "id" and "type". If supplied, the removed duplicates are reported per node-type pair.
directed	Boolean. If FALSE, A-B and B-A are the same edge. Defaults to FALSE.
weight_agg	String. How to combine the weights of duplicates. One of c("max", "sum", "mean"). Ignored without a "weight" column. Defaults to "max", matching the deduplication in node2vec() .
.verbose	Boolean. Controls verbosity. Defaults to TRUE.

Value

A data.table with the columns "from", "to" and, if present in the input, "weight". All other columns are dropped. For undirected graphs each edge is stored once with from <= to (lexicographically). Endpoints are returned as character.

differential_graph_test_data

Generate Synthetic Differential Graph Data

Description

Creates a pair of graphs with known topological differences to verify context-aware differential embedding analysis. The two graphs share a common backbone but differ in defined regions:

- Community 1 is a stable negative control, identical across both graphs.
- Community 2 contains a hub node demoted to a peripheral node in graph 2.
- Two bridge nodes span communities 2 and 3 in graph 1 but are fully embedded within community 3 in graph 2.
- A set of exclusive nodes appears in only one graph.

Usage

```
differential_graph_test_data(
  n_stable = 50L,
  n_comm2 = 50L,
  n_comm3 = 50L,
  n_exclusive = 5L
)
```

Arguments

n_stable	Integer. Number of nodes in the stable negative-control community.
n_comm2	Integer. Number of regular nodes in community 2, excluding the hub.
n_comm3	Integer. Number of nodes in community 3, excluding bridge nodes.
n_exclusive	Integer. Number of exclusive nodes per graph.

Value

A named list containing:

- g1_edges - data.table with columns from, to for graph 1.
- g1_nodes - data.table with columns node, cluster for graph 1.
- g2_edges - data.table with columns from, to for graph 2.
- g2_nodes - data.table with columns node, cluster for graph 2.
- data_info - data.table with columns node, is_differential, node_status ("shared", "g1_only", "g2_only").

diffuse_scores	<i>Run diffusion scoring</i>
----------------	------------------------------

Description

Diffuses input scores over the precomputed kernel. Requires build_kernel() to have been called first.

Usage

```
diffuse_scores(
  object,
  input,
  background = NULL,
  method = "raw",
  n_perm = 1000L,
  seed = 42L
)
```

Arguments

object	A DiffusionScores object with a built kernel.
input	Named numeric vector or named matrix. Names must correspond to node names in the graph. For a vector, treated as a single input column. For a matrix, rows are background nodes and columns are separate input sets.
background	Character vector. Node names forming the background set. Defaults to all nodes in input.
method	Character. One of "raw", "z", or "mc". Defaults to "raw".
n_perm	Integer. Number of permutations for method = "mc". Defaults to 1000L.
seed	Integer. RNG seed for method = "mc". Defaults to 42L.

Value

The DiffusionScores object with scores populated as a `data.table` with columns `node` and one score column per input.

DiffusionProfiles	<i>DiffusionProfiles</i>
-------------------	--------------------------

Description

S7 class for the diffusion profiles of Ruiz et al.: one constrained personalised PageRank per seed over a heterogeneous graph. The seed is a source; every other node of a sink type absorbs the walker. With type weights, the walker at node *i* first picks a neighbouring node type proportional to its weight, then a neighbour of that type.

Usage

```
DiffusionProfiles(
  graph_dt,
  node_dt,
  sink_types,
  type_weights = NULL,
  directed = FALSE
)
```

Arguments

<code>graph_dt</code>	<code>data.table</code> . The edge table. Needs the columns "from" and "to", optionally "weight" (non-negative). Every endpoint needs to be in <code>node_dt\$id</code> .
<code>node_dt</code>	<code>data.table</code> . The node table with the columns "id" and "type".
<code>sink_types</code>	Character vector. Node types that absorb the walker, e.g. <code>c("drug", "disease")</code> . Can be empty.
<code>type_weights</code>	Optional named numeric vector. Positive weight per node type; needs to cover every type in <code>node_dt</code> . NULL gives the plain (edge-weighted) random walk.
<code>directed</code>	Boolean. Treat the graph as directed. Defaults to FALSE.

Value

An initialised DiffusionProfiles object.

Properties

graph_dt `data.table`. The edge table with from, to and optionally weight.
node_dt `data.table`. The node table with id and type.
sink_types Character vector. Node types that act as sinks.
type_weights Named numeric vector of weights per node type, or NULL for the plain random walk.
profiles Numeric matrix of nodes x seeds. NULL until `generate_profiles()` is called.
params Named list of the parameters used.

References

Ruiz, Zitnik and Leskovec, Identification of disease treatment mechanisms through the multiscale interactome, Nat Commun 2021.

DiffusionScores	<i>DiffusionScores</i>
-----------------	------------------------

Description

S7 class wrapping the Rust spectral diffusion backend. Holds the input graph, the precomputed kernel (as an external pointer), and the resulting per-node diffusion scores.

Usage

```
DiffusionScores(graph)
```

Arguments

graph An igraph object.

Value

An initialised DiffusionScores object.

Properties

graph An igraph object. The input graph.

kernel_pointer External pointer to the Rust DiffusionKernel. NULL until build_kernel() is called.

scores A data.table of diffusion results. Empty until diffuse() is called.

params Named list of parameters used during kernel construction and diffusion.

download_database	<i>Download database from GitHub release</i>
-------------------	--

Description

Download database from GitHub release

Usage

```
download_database()
```

EmbedDrift

EmbedDrift

Description

S7 class for context-aware differential node embedding analysis between two graphs. Holds both input graphs, their embeddings, the Procrustes-aligned embedding, and the resulting cosine similarity statistics.

Usage

```
EmbedDrift(graph_dt_1, graph_dt_2)
```

Arguments

<code>graph_dt_1</code>	A data.frame with columns <code>from</code> , <code>to</code> , and optionally <code>weight</code> representing the first graph.
<code>graph_dt_2</code>	A data.frame with columns <code>from</code> , <code>to</code> , and optionally <code>weight</code> representing the second graph.

Value

An initialised EmbedDrift object.

Properties

graph_dt_1 A data.frame containing the edges of graph 1. Must have columns `from`, `to`, and optionally `weight`.

graph_dt_2 A data.frame containing the edges of graph 2. Must have columns `from`, `to`, and optionally `weight`.

embd_1 Numeric matrix. Node embedding for graph 1, rows are nodes.

embd_2 Numeric matrix. Node embedding for graph 2, rows are nodes.

aligned_embd Numeric matrix. Procrustes-aligned embedding of graph 1 onto graph 2, restricted to shared nodes.

stats A data.frame holding per-node cosine similarity scores and node metadata after running the differential analysis.

params Named list of parameters used during embedding and alignment.

 evaluate_node2vec_test

Evaluate node2vec test results

Description

Compares embedding similarities against ground truth clusters.

Usage

```
evaluate_node2vec_test(embeddings, node_labels)
```

Arguments

embeddings	Matrix of node embeddings (nodes x dimensions).
node_labels	data.table with columns: node, cluster.

Value

A list with:

- within_cluster_sim - Mean similarity within clusters.
- between_cluster_sim - Mean similarity between clusters.
- within_dist - The within cluster similarities as vector.
- between_dist - The between cluster similarities as vector.

 generate_initial_emb

Generate the initial graph embedding

Description

This function will generate the initial node2vec embedding with the provided parameters. In brief, it will generate random walks over the graph and fit a Skip Gram model for the number of defined epochs to generate on a per node basis an embedding vector. These will be subsequently used for the calculation of test statistics. For more details, please refer to Ietswaart, et al.

Usage

```
generate_initial_emb(
  object,
  embd_dim = 8L,
  n_graph = 3L,
  genewalk_params = params_genewalk(),
  directed = FALSE,
  seed = 42L,
  .verbose = TRUE
)
```

Arguments

object	The GeneWalk class, please see GeneWalk() .
embd_dim	Integer. Size of the embedding dimensions to create. Defaults to 8L in line with the authors recommendations.
n_graph	Integer. How many initial graphs to create. Defaults to 3L.
genewalk_params	Named list. Contains the node2vec parameters. The list has the following elements: • p - Numeric. Return parameter for biased random walks. Defaults to 1.0. • q - Numeric. In-out parameter for biased random walks. Defaults to 1.0. • walks_per_node - Integer. Number of random walks per node. Defaults to 100L. • walk_length - Integer. Length of each random walk. Defaults to 10L. • num_workers - Number of worker threads during to use during SGD updates. For determinism, defaults to 1L. • batch_size - Integer. Batch size for training. Defaults to 256L. • n_epochs - Integer. Number of training epochs. Defaults to 15L. • n_negatives - Integer. Number of negative samples. Defaults to 5L. • window_size - Integer. Context window size. Defaults to 2L. • lr - Numeric. Learning rate. Defaults to 1e-2.
directed	Boolean. Indicates if this is a directed or undirected network. Defaults to FALSE.
seed	Integer. Seed for reproducibility.
.verbose	Boolean. Controls verbosity of the function

Value

The object with the added initial embeddings.

References

Ietswaart, et al., Genome Biol, 2021

generate_initial_embeddings

Generate initial node embeddings for both graphs

Description

Runs node2vec independently on both graphs using identical hyperparameters, storing the resulting embedding matrices in the object. Graph 1 serves as the reference for subsequent Procrustes alignment. For details on the node2vec algorithm, see Grover & Leskovec. For the approach generally speaking, see Amir, et al.

Usage

```
generate_initial_embeddings(
  object,
  embd_dim = 64L,
  node2vec_params = params_node2vec(),
  directed = FALSE,
  seed = 42L,
  .verbose = TRUE
)
```

Arguments

object	An EmbedDrift object, see EmbedDrift() .
embd_dim	Integer. Embedding dimension size. Defaults to 64L.
node2vec_params	Named list. Node2vec training parameters, see params_node2vec() .
directed	Boolean. Whether to treat both graphs as directed. Defaults to FALSE.
seed	Integer. Random seed for reproducibility.
.verbose	Boolean. Controls verbosity. Defaults to TRUE.

Value

The EmbedDrift object with embd_1, embd_2, and params populated.

References

Grover & Leskovec, arXiv, 2016
 Amir, et al., Cancer Cell, 2025

generate_permuted_emb *Generate the random, permuted embedding*

Description

This function will generate generate permuted subgraphs that show the same node degree and run the node2vec algorithm on top of these subgraphs. The cosine distance between the real edges will be calculated and stored in the object. These will be used subsequently for statistical significance testing. For more details, please refer to Ietswaart, et al. The function will return a warning and the class as is if you have not run [generate_initial_emb\(\)](#).

Usage

```
generate_permuted_emb(object, n_perm = 3L, seed = 42L, .verbose = TRUE)
```

Arguments

object	The GeneWalk class, please see GeneWalk() .
n_perm	Inter. Number of permutations to generate. Defaults to 3L, the recommendation of the authors.
seed	Integer. Seed for reproducibility.
.verbose	Boolean. Controls verbosity of the function

Details

The function will be using the same parameters for the node2vec training as were used for [generate_initial_emb\(\)](#).

Value

The object with the added random embeddings for statistical testing.

References

Ietswaart, et al., Genome Biol, 2021

generate_profiles	<i>Generate diffusion profiles</i>
-------------------	------------------------------------

Description

Runs one constrained personalised PageRank per seed (Ruiz et al.). The graph is built once in Rust and the seeds run in parallel. Each call replaces the stored profiles, so pass drugs and diseases together if you want to compare them afterwards.

Usage

```
generate_profiles(
  object,
  seeds,
  diffusion_profile_params = params_diffusion_profiles(),
  .verbose = TRUE
)
```

Arguments

object	A DiffusionProfiles object, see DiffusionProfiles() .
seeds	Character vector. Node ids to start the walks from; one profile per seed.
diffusion_profile_params	Named list. The PageRank parameters, see params_diffusion_profiles() .
.verbose	Boolean. Controls verbosity. Defaults to TRUE.

Value

The DiffusionProfiles object with profiles populated as a numeric matrix of nodes x seeds. Each column sums to 1.

GeneWalk	<i>GeneWalk</i>
----------	-----------------

Description

Class that stores keeps all of the important data for the Gene Walk algorithm, see Ietswaart, et al. The class can be subsequently used for to apply the Gene Walk algorithm.

Usage

```
GeneWalk(graph_dt, gene_to_pathway_dt, gene_ids, pathway_ids)
```

Arguments

graph_dt	A data.table with the graph information. Needs to have the columns "from", "to" and "type"; type must include "hierarchy" for the pathways, "interaction" for interactions and "part_of" for gene/pathway interactions, and optionally a weight.
gene_to_pathway_dt	A data.table with the gene to pathway information. Needs to have "from" (gene) and "to" (pathway).
gene_ids	String. The "bag of genes" you are testing.
pathway_ids	String. The pathways represented in the network.

Value

Returns the initialised GeneWalk initialised object for subsequent analysis.

Properties

graph_dt A data.table that contains the initial graph.

gene_to_pathway_dt A data.table that contains the gene to pathway connections.

gene_ids The genes represented in your gene walk list.

pathway_ids The pathways represented in your gene walk network.

embds The initial embedding matrices generated by node2vec on top of the graph. Depends on the n_graph parameter you choose.

permuted_embd A list containing cosine distances of permuted graphs..

stats A data.table containing the information on the statistical testing across the permutations.

params A list in which used parameters will be stored.

References

Ietswaart, et al., Genome Biol, 2021

GeneWalkGenerator	<i>Gene Walk Network Generator</i>
-------------------	------------------------------------

Description

R6 class to build Gene Walk networks from internal database sources. Supports building a master network once and subsetting to multiple gene sets.

Public fields

pathway_sources Character vector of pathway sources
 pathway_namespaces List of namespace filters per pathway source
 ppi_sources Character vector of PPI sources
 ppi_params List of PPI filtering parameters

Methods

Public methods:

- `GeneWalkGenerator$new()`
- `GeneWalkGenerator$print()`
- `GeneWalkGenerator$add_pathways()`
- `GeneWalkGenerator$add_ppi()`
- `GeneWalkGenerator$build()`
- `GeneWalkGenerator$create_for_genes()`
- `GeneWalkGenerator$reset_choices()`
- `GeneWalkGenerator$return_full_network_dt()`
- `GeneWalkGenerator$clone()`

`GeneWalkGenerator$new()`: Initialise the generator

Usage:

```
GeneWalkGenerator$new()
```

`GeneWalkGenerator$print()`: Print for GeneWalkGenerator

Usage:

```
GeneWalkGenerator$print(...)
```

Arguments:

... Additional parameters to forward to the print

`GeneWalkGenerator$add_pathways()`: Add pathway sources to the network

Usage:

```
GeneWalkGenerator$add_pathways(
  source = c("go", "reactome"),
  go_namespace = c("biological_process", "molecular_function", "cellular_component")
)
```

Arguments:

source Character vector of sources: "go", "reactome". Defaults to "go" only by default.
 go_namespace Character vector. Specifically for GO. Choices are c("biological_process",
 "molecular_function", "cellular_component")

GeneWalkGenerator\$add_ppi(): Add PPI sources to the network

Usage:

```
GeneWalkGenerator$add_ppi(
  source = c("combined", "string", "signor", "reactome", "intact"),
  string_threshold = NULL,
  intact_threshold = NULL,
  intact_physical_only = FALSE
)
```

Arguments:

source Character vector of sources
 string_threshold Numeric threshold for STRING scores
 intact_threshold Numeric threshold for Intact scores
 intact_physical_only Logical, only physical interactions from Intact

GeneWalkGenerator\$build(): Build the full network from selected sources

Usage:

```
GeneWalkGenerator$build(.verbose = TRUE)
```

Arguments:

.verbose Boolean. Controls the verbosity of the function

GeneWalkGenerator\$create_for_genes(): Create a gene-specific GeneWalk object

Usage:

```
GeneWalkGenerator$create_for_genes(genes)
```

Arguments:

genes Character vector of gene symbols

Returns: Returns the initialised GeneWalk.

GeneWalkGenerator\$reset_choices(): Resets the internal choices and erases any stored network data.

Usage:

```
GeneWalkGenerator$reset_choices()
```

GeneWalkGenerator\$return_full_network_dt(): Returns the full network data.table.

Usage:

```
GeneWalkGenerator$return_full_network_dt()
```

Returns: data.table with the full internal network.

GeneWalkGenerator\$clone(): The objects of this class are cloneable with this method.

Usage:

```
GeneWalkGenerator$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

get_aligned_embd	<i>Get the aligned embedding</i>
------------------	----------------------------------

Description

Get the aligned embedding

Usage

get_aligned_embd(object)

Arguments

object An EmbedDrift object, see [EmbedDrift\(\)](#).

Value

A numeric matrix of the Procrustes-aligned embedding of graph 1 onto graph 2, restricted to shared nodes.

get_db_connection	<i>Internal connector to the DB</i>
-------------------	-------------------------------------

Description

Downloads the DuckDB if not available and unzips it.

Usage

get_db_connection()

Value

The DuckDB connection to the internal data

get_embedding	<i>Get the list of embeddings</i>
---------------	-----------------------------------

Description

Get the list of embeddings

Usage

```
get_embedding(object)
```

Arguments

object	The GeneWalk class, please see GeneWalk() .
--------	---

Value

If found, returns the embedding list.

get_embeddings	<i>Get node embeddings</i>
----------------	----------------------------

Description

Get node embeddings

Usage

```
get_embeddings(object, which = NULL)
```

Arguments

object	An EmbedDrift object, see EmbedDrift() .
which	Integer. Which embedding to return: 1, 2, or NULL for both as a named list. Defaults to NULL.

Value

A matrix, or a named list with elements embd_1 and embd_2.

get_gene_info	<i>Get the gene main information</i>
---------------	--------------------------------------

Description

Get the gene main information

Usage

```
get_gene_info(head_only = FALSE)
```

Arguments

head_only Logical. If TRUE, returns only the first 5 rows.

Details

The data is extracted from the OpenTargets release 26.03

Value

data.table with the gene information

get_gene_ontology_hierarchy	<i>Get the Gene Ontology</i>
-----------------------------	------------------------------

Description

Get the Gene Ontology

Usage

```
get_gene_ontology_hierarchy(head_only = FALSE)
```

Arguments

head_only Logical. If TRUE, returns only the first 5 rows.

Details

The data is extracted from the obo file downloaded from [GO OBO Foundry](#). Last download: March '26.

Value

data.table with the GO to GO edges with edge type info.

`get_gene_ontology_info`*Get the Gene Ontology main information*

Description

Get the Gene Ontology main information

Usage

```
get_gene_ontology_info(head_only = FALSE)
```

Arguments

`head_only` Logical. If TRUE, returns only the first 5 rows.

Details

The data is extracted from the obo file downloaded from [GO OBO Foundry](#). Last download: March '26.

Value

data.table with the Gene Ontology information

`get_gene_pathway_dt`*Get the gene to pathway data.table*

Description

Get the gene to pathway data.table

Usage

```
get_gene_pathway_dt(object)
```

Arguments

`object` The GeneWalk class, please see [GeneWalk\(\)](#).

Value

Returns the gene to pathway data.table stored in the object

get_gene_to_go	<i>Get the Gene Ontology to gene connections</i>
----------------	--

Description

Get the Gene Ontology to gene connections

Usage

```
get_gene_to_go(head_only = FALSE)
```

Arguments

head_only Logical. If TRUE, returns only the first 5 rows.

Details

The data is extracted from the OpenTargets release 26.03

Value

data.table with the Gene Ontology to gene edges

get_gene_to_reactome	<i>Get the Reactome to gene connections</i>
----------------------	---

Description

Get the Reactome to gene connections

Usage

```
get_gene_to_reactome(head_only = FALSE)
```

Arguments

head_only Logical. If TRUE, returns only the first 5 rows.

Details

The data is extracted from the OpenTargets release 26.03

Value

data.table with the Reactome to gene edges

get_graph_dt	<i>Get the edge data.table</i>
--------------	--------------------------------

Description

Get the edge data.table

Usage

```
get_graph_dt(object)
```

Arguments

object The GeneWalk class, please see [GeneWalk\(\)](#).

Value

Returns the edge data.table stored in the object

get_graph_dts	<i>Get graph edge tables</i>
---------------	------------------------------

Description

Get graph edge tables

Usage

```
get_graph_dts(object, which = NULL)
```

Arguments

object An EmbedDrift object, see [EmbedDrift\(\)](#).
which Integer. Which graph table to return: 1, 2, or NULL for both as a named list.
Defaults to NULL.

Value

A data.frame, or a named list with elements graph_dt_1 and graph_dt_2.

get_gw_data	<i>Get the full GW data stored in the class</i>
-------------	---

Description

Get the full GW data stored in the class

Usage

```
get_gw_data(x)

## S3 method for class 'DataBuilder'
get_gw_data(x)
```

Arguments

x An object from which to extract the genewalk data.

Value

A data.table with the full data stored in the class.

get_gw_data_filtered	<i>Get the filtered GW data for a list of genes</i>
----------------------	---

Description

Get the filtered GW data for a list of genes

Usage

```
get_gw_data_filtered(x, gene_ids)

## S3 method for class 'DataBuilder'
get_gw_data_filtered(x, gene_ids)
```

Arguments

x An object from which to extract the genewalk data
gene_ids String. The gene ids to which to filter the data.

Value

A list with the following elements

- gwn - The final GeneWalk network.
- genes_to_pathways - The genes to pathway information.
- represented_genes - All included genes.
- represented_pathways - All included pathways.

`get_interactions_combined`*Get the Combined network*

Description

This returns a combined network akin to the workflow from Barrio-Hernandez et al. based on high quality edges from STRING, physical interactions from Intact and the full sets from SIGNOR, Reactome and Pathway Commons interactions.

Usage

```
get_interactions_combined(head_only = FALSE)
```

Arguments

`head_only` Logical. If TRUE, returns only the first 5 rows.

Details

The data is extracted from the OpenTargets release 26.03

Value

data.table with the combined, collapsed network data.

References

Barrio-Hernandez et al., Nat Gen, 2023

`get_interactions_intact`*Get the Intact network*

Description

Get the Intact network

Usage

```
get_interactions_intact(  
  threshold = NULL,  
  physical_interactions = FALSE,  
  head_only = FALSE  
)
```

Arguments

threshold	Optional numeric. If you wish to filter on the STRING score.
physical_interactions	Boolean. Shall only physical interactions be returned.
head_only	Logical. If TRUE, returns only the first 5 rows.

Details

The data is extracted from the OpenTargets release 26.03

Value

data.table with the Intact network data.

`get_interactions_pc`*Get the pathway commons network*

Description

Get the pathway commons network

Usage

```
get_interactions_pc(head_only = FALSE)
```

Arguments

head_only	Logical. If TRUE, returns only the first 5 rows.
-----------	--

Details

This is version 14 of Pathway Commons. For details, check Rodchenkov, et al., Nucleic Acids Res., 2019

Value

data.table with the Pathway Commons network data.

```
get_interactions_reactome
```

Get the Reactome network

Description

Get the Reactome network

Usage

```
get_interactions_reactome(head_only = FALSE)
```

Arguments

head_only Logical. If TRUE, returns only the first 5 rows.

Details

The data is extracted from the OpenTargets release 26.03

Value

data.table with the Reactome network data.

```
get_interactions_signor
```

Get the SIGNOR network

Description

Get the SIGNOR network

Usage

```
get_interactions_signor(head_only = FALSE)
```

Arguments

head_only Logical. If TRUE, returns only the first 5 rows.

Details

The data is extracted from the OpenTargets release 26.03

Value

data.table with the SIGNOR network data.

get_interactions_string	<i>Get the STRING network</i>
-------------------------	-------------------------------

Description

Get the STRING network

Usage

get_interactions_string(threshold = NULL, head_only = FALSE)

Arguments

- threshold Optional numeric. If you wish to filter on the STRING score.
- head_only Logical. If TRUE, returns only the first 5 rows.

Details

The data is extracted from the OpenTargets release 26.03

Value

data.table with the String network data.

get_profiles	<i>Get diffusion profiles</i>
--------------	-------------------------------

Description

Get diffusion profiles

Usage

get_profiles(object)

Arguments

- object An object containing diffusion profiles.

Value

A numeric matrix of nodes x seeds.

`get_reactome_hierarchy`*Get the Reactome pathway hierarchy*

Description

Get the Reactome pathway hierarchy

Usage

```
get_reactome_hierarchy(  
  relationship = c("child_of", "descendant_of", "parent_of", "ancestor_of"),  
  head_only = FALSE  
)
```

Arguments

relationship	String vector. Defines which relationships to return. Multiple allowed and choices from c("child_of", "descendant_of", "parent_of", "ancestor_of")
head_only	Logical. If TRUE, returns only the first 5 rows.

Details

The data is extracted from the OpenTargets release 26.03

Value

data.table with the Reactome to Reactome edges with edge type info.

`get_reactome_info`*Get the Reactome main information*

Description

Get the Reactome main information

Usage

```
get_reactome_info(head_only = FALSE)
```

Arguments

head_only	Logical. If TRUE, returns only the first 5 rows.
-----------	--

Details

The data is extracted from the OpenTargets release 26.03

Value

data.table with the Reactome information

get_scores	<i>Get scores</i>
------------	-------------------

Description

Get scores

Usage

get_scores(object)

Arguments

object An object containing scores.

Value

A data.table of scores.

get_stats	<i>Get the statistical results</i>
-----------	------------------------------------

Description

Get the statistical results

Usage

get_stats(object)

Arguments

object The GeneWalk (see [GeneWalk\(\)](#)) or EmbedDrift class (see [EmbedDrift\(\)](#)).

Value

If found, the statistics results within the class

metapath2vec

Metapath2Vec

Description

Runs metapath2vec over a heterogeneous graph. Random walks are constrained to follow a meta-path schema over node types, e.g. `c("gene", "pathway", "gene")`, and a skip-gram model is trained on them. With `metapath_plus = TRUE`, negative samples are drawn from the context node's own type (`metapath2vec++`).

Walks start only on nodes of the metapath's first type. Nodes no walk ever reaches keep their random initialisation, so by default they are removed and listed in the `unvisited_nodes` attribute. The function warns when more than 10% of walks are dropped or walks reach less than half the requested length on average: the schema does not fit the graph. Details are in the `walk_stats` attribute. A metapath over a single type warns as well; that is DeepWalk on a subgraph, so use [node2vec\(\)](#).

Usage

```
metapath2vec(
  graph_dt,
  node_dt,
  metapath,
  embd_dim = 8L,
  metapath2vec_params = params_metapath2vec(),
  metapath_plus = FALSE,
  filter_unvisited = TRUE,
  directed = FALSE,
  seed = 42L,
  .verbose = TRUE
)
```

Arguments

<code>graph_dt</code>	data.table. The edge table. Needs to have the columns "from" and "to", and can optionally have a "weight" column.
<code>node_dt</code>	data.table. The node table with the columns "id" and "type". Every edge endpoint needs to be in id.
<code>metapath</code>	Character vector. The metapath over node types, closing on its starting type, e.g. <code>c("gene", "pathway", "gene")</code> .
<code>embd_dim</code>	Integer. Size of the embedding dimensions to create. Defaults to 8L.
<code>metapath2vec_params</code>	Named list. The training parameters, see params_metapath2vec() .
<code>metapath_plus</code>	Boolean. Use per-type negative sampling (<code>metapath2vec++</code>). Defaults to FALSE.
<code>filter_unvisited</code>	Boolean. Remove the rows of nodes that no walk visited. Defaults to TRUE.

directed	Boolean. Indicates if this is a directed or undirected network. Defaults to FALSE.
seed	Integer. Seed for reproducibility.
.verbose	Boolean. Controls verbosity of the function.

Value

A numeric matrix of n_nodes x embd_dim with the node ids as rownames. Rows are ordered by node type. Carries three attributes:

- node_type - Named factor with the type of each row.
- unvisited_nodes - Character vector with the ids of the nodes no walk visited. Removed from the matrix if filter_unvisited = TRUE.
- walk_stats - List with start_nodes, attempted, truncated, dropped, mean_length and walk_length.

References

Dong, Chawla and Swami, metapath2vec: Scalable Representation Learning for Heterogeneous Networks, KDD 2017.

myc_genes	<i>MYC target genes</i>
-----------	-------------------------

Description

The MYC target genes from the MSigDB (Hallmarks V1).

Usage

myc_genes

Format

myc_genes:
A data.table with 200 rows and 3 columns:
ensembl_gene Ensembl identifier
gene_symbol Gene Symbol
gene_set Name of the gene set

References

Liberzon, et al., Cell Syst, 2015

n_nodes	<i>Number of nodes represented</i>
---------	------------------------------------

Description

Number of nodes represented

Usage

```
n_nodes(x)

## S3 method for class 'DataBuilder'
n_nodes(x)
```

Arguments

x An object from which to get the number of nodes.

Value

Number of unique nodes stored in the class.

new_data_builder	<i>Generate a new DataBuilder helper class</i>
------------------	--

Description

Generate a new DataBuilder helper class

Usage

```
new_data_builder(ppis, gene_to_pathways, pathway_hierarchy)
```

Arguments

ppis data.table. The full interaction data to subset subsequently.
gene_to_pathways data.table. The gene to pathway associations.
pathway_hierarchy data.table. The pathway ontology

Value

Returns the initialised DataBuilder object.

node2vec

*Node2Vec***Description**

Runs the Node2Vec algorithm

Usage

```
node2vec(
  graph_dt,
  embd_dim = 8L,
  node2vec_params = params_node2vec(),
  directed = FALSE,
  seed = 42L,
  .verbose = TRUE
)
```

Arguments

graph_dt	data.table. The edge data.frame. Needs to have the columns "from" and "to". Can also optionally a "weight" column.
embd_dim	Integer. Size of the embedding dimensions to create. Defaults to 8L in line with the authors recommendations.
node2vec_params	Named list. Contains the node2vec parameters. The list has the following elements: • p - Numeric. Return parameter for biased random walks. Defaults to 1.0. • q - Numeric. In-out parameter for biased random walks. Defaults to 1.0. • walks_per_node - Integer. Number of random walks per node. Defaults to 40L. • walk_length - Integer. Length of each random walk. Defaults to 40L. • num_workers - Number of worker threads during batching. Defaults to 4L. To note: during graph generation, Rust will use via the Rayon backend all available threads. • batch_size - Integer. Batch size for training. Defaults to 256L. • n_epochs - Integer. Number of training epochs. Defaults to 15L. • n_negatives - Integer. Number of negative samples. Defaults to 5L. • window_size - Integer. Context window size. Defaults to 2L. • lr - Numeric. Learning rate. Defaults to 1e-2.
directed	Boolean. Indicates if this is a directed or undirected network. Defaults to FALSE.
seed	Integer. Seed for reproducibility.
.verbose	Boolean. Controls verbosity of the function

node2vec_test_data	<i>Generate Simple Node2Vec Test Data</i>
--------------------	---

Description

Creates graphs with known structure to verify node2vec is working correctly. The following options exists:

- "barbell" - Two dense clusters connected by single edge.
- "cavemen" - Multiple isolated cliques with sparse inter-connections.
- "stochastic_block" - Communities with controlled edge probabilities.

Usage

```
node2vec_test_data(
  test_type = c("barbell", "cavemen", "stochastic_block"),
  n_nodes_per_cluster = 10L,
  n_clusters = 3L,
  p_within = 0.8,
  p_between = 0.05,
  seed = 42L
)
```

Arguments

test_type	String. One of c("barbell", "cavemen", "stochastic_block")
n_nodes_per_cluster	Integer. Nodes per cluster/community.
n_clusters	Integer. Number of clusters.
p_within	Numeric. Probability of edges within cluster (0-1).
p_between	Numeric. Probability of edges between clusters (0-1).
seed	Integer. Random seed.

Value

A list containing:

edges	data.table with from, to, edge_type
node_labels	data.table with node, cluster

params_diffusion_profiles

Wrapper function for the diffusion profile parameters

Description

Parameters for `generate_profiles()`, i.e. the constrained personalised PageRank behind the diffusion profiles of Ruiz et al.

Usage

```
params_diffusion_profiles(alpha = 0.85, max_iter = 100L, tol = 1e-06)
```

Arguments

alpha	Numeric. Probability of continuing the walk instead of restarting at the seed. Defaults to 0.85.
max_iter	Integer. Maximum number of power iterations. Defaults to 100L.
tol	Numeric. Convergence threshold on the L1 change between two iterations. Defaults to 1e-06.

Value

A named list with the following elements:

- alpha - Numeric. Probability of continuing the walk instead of restarting at the seed. Defaults to 0.85.
- max_iter - Integer. Maximum number of power iterations. Defaults to 100L.
- tol - Numeric. Convergence threshold on the L1 change between two iterations. Defaults to 1e-06.

params_genewalk

Wrapper function for the GeneWalk-specific node2vec parameters

Description

Wrapper function for the GeneWalk-specific node2vec parameters

Usage

```
params_genewalk(
  p = 1,
  q = 1,
  walks_per_node = 100L,
  walk_length = 10L,
  batch_size = 256L,
  n_epochs = 5L,
  n_negatives = 5L,
  window_size = 1L,
  lr = 0.01,
  num_workers = 1L
)
```

Arguments

p	Numeric. Return parameter for biased random walks. Defaults to 1.0.
q	Numeric. In-out parameter for biased random walks. Defaults to 1.0.
walks_per_node	Integer. Number of random walks per node. Defaults to 100L.
walk_length	Integer. Length of each random walk. Defaults to 10L.
batch_size	Integer. Batch size for training. Defaults to 256L.
n_epochs	Integer. Number of training epochs. Defaults to 5L.
n_negatives	Integer. Number of negative samples. Defaults to 5L.
window_size	Integer. Context window size. For GeneWalk you want embeddings from direct neighbours. Defaults to 1L.
lr	Numeric. Learning rate. Defaults to 0.01.
num_workers	Integer. Number of worker threads. Anything above 1L makes the fitting much faster, at the cost of reproducibility. Defaults to 1L.

Value

A named list with the following elements:

- p - Numeric. Return parameter for biased random walks. Defaults to 1.0.
- q - Numeric. In-out parameter for biased random walks. Defaults to 1.0.
- walks_per_node - Integer. Number of random walks per node. Defaults to 100L.
- walk_length - Integer. Length of each random walk. Defaults to 10L.
- num_workers - Integer. Number of worker threads. Anything above 1L makes the fitting much faster, at the cost of reproducibility. Defaults to 1L.
- batch_size - Integer. Batch size for training. Defaults to 256L.
- n_epochs - Integer. Number of training epochs. Defaults to 5L.
- n_negatives - Integer. Number of negative samples. Defaults to 5L.
- window_size - Integer. Context window size. For GeneWalk you want embeddings from direct neighbours. Defaults to 1L.
- lr - Numeric. Learning rate. Defaults to 0.01.

params_genewalk_data *GeneWalk synthetic data params*

Description

GeneWalk synthetic data params

Usage

```
params_genewalk_data(
  n_signal_genes = 150L,
  n_noise_genes = 150L,
  n_roots = 10L,
  depth = 3L,
  branching = 3L,
  p_lateral = 0.1,
  p_ppi = 0.05,
  min_annotations = 10L,
  max_annotations = 25L,
  min_noise_subtrees = 15L
)
```

Arguments

n_signal_genes	Integer. Number of signal genes. Defaults to 150L.
n_noise_genes	Integer. Number of noise genes. Defaults to 150L.
n_roots	Integer. Number of ontology root nodes. Defaults to 10L.
depth	Integer. Depth of each ontology subtree. Defaults to 3L.
branching	Integer. Base branching factor per ontology node. Defaults to 3L.
p_lateral	Numeric. Probability of a lateral edge between sibling nodes. Defaults to 0.1.
p_ppi	Numeric. Probability of a PPI edge between any two genes in the same group. Defaults to 0.05.
min_annotations	Integer. Minimum ontology term annotations per gene. Defaults to 10L.
max_annotations	Integer. Maximum ontology term annotations per gene. Defaults to 25L.
min_noise_subtrees	Integer. Minimum distinct ontology subtrees a noise gene must span. Defaults to 15L.

Value

A named list with the following elements:

- n_signal_genes - Integer. Number of signal genes. Defaults to 150L.

- n_noise_genes - Integer. Number of noise genes. Defaults to 150L.
- n_roots - Integer. Number of ontology root nodes. Defaults to 10L.
- depth - Integer. Depth of each ontology subtree. Defaults to 3L.
- branching - Integer. Base branching factor per ontology node. Defaults to 3L.
- p_lateral - Numeric. Probability of a lateral edge between sibling nodes. Defaults to 0.1.
- p_ppi - Numeric. Probability of a PPI edge between any two genes in the same group. Defaults to 0.05.
- min_annotations - Integer. Minimum ontology term annotations per gene. Defaults to 10L.
- max_annotations - Integer. Maximum ontology term annotations per gene. Defaults to 25L.
- min_noise_subtrees - Integer. Minimum distinct ontology subtrees a noise gene must span. Defaults to 15L.

params_kernel

Wrapper function for the diffusion kernel parameters

Description

These are parameters for the graph diffusion kernels. Not all parameters are used by all kernels, please see Details.

Usage

```
params_kernel(sigma2 = 1, add_diag = 1, a = 3, p = 5L)
```

Arguments

sigma2	Numeric. Bandwidth parameter controlling the extent of diffusion. Used by regularised_laplacian and diffusion kernels. Defaults to 1.0.
add_diag	Numeric. Regularisation term added to the diagonal. Used by regularised_laplacian. Must be positive for every connected component to be invertible. Defaults to 1.0.
a	Numeric. Regularisation parameter for the pstep kernel. Must be ≥ 2 , since the eigenvalues of the normalised Laplacian sit in $[0, 2]$. Defaults to 3.0.
p	Integer. Number of steps for the pstep kernel. Must be positive. Defaults to 5L.

Details

Parameter usage by kernel type:

- regularised_laplacian: sigma2, add_diag
- commute_time: none
- diffusion: sigma2
- inverse_cosine: none
- pstep: a, p

Value

A named list with the following elements:

- sigma2 - Numeric. Bandwidth parameter controlling the extent of diffusion. Used by regularised_laplacian and diffusion kernels. Defaults to 1.0.
- add_diag - Numeric. Regularisation term added to the diagonal. Used by regularised_laplacian. Must be positive for every connected component to be invertible. Defaults to 1.0.
- a - Numeric. Regularisation parameter for the pstep kernel. Must be ≥ 2 , since the eigenvalues of the normalised Laplacian sit in $[0, 2]$. Defaults to 3.0.
- p - Integer. Number of steps for the pstep kernel. Must be positive. Defaults to 5L.

params_metapath2vec	<i>Wrapper function for the metapath2vec parameters</i>
---------------------	---

Description

Parameters for `metapath2vec()`. There is no p or q: metapath walks are first-order and follow the schema, not a biased return/explore rule.

Usage

```
params_metapath2vec(
  walks_per_node = 40L,
  walk_length = 40L,
  n_epochs = 20L,
  n_negatives = 5L,
  window_size = 2L,
  lr = 0.01,
  sample = 0.001,
  num_workers = NULL
)
```

Arguments

walks_per_node	Integer. Number of random walks per node of the metapath's starting type. Defaults to 40L.
walk_length	Integer. Length of each random walk. Rounded up so the walk closes on a full number of schema cycles. Defaults to 40L.
n_epochs	Integer. Number of training epochs. Defaults to 20L.
n_negatives	Integer. Number of negative samples. Defaults to 5L.
window_size	Integer. Context window size. Defaults to 2L.
lr	Numeric. Learning rate. Defaults to 0.01.
sample	Numeric. Subsampling threshold for frequent nodes. Defaults to 0.001.
num_workers	Integer or NULL. Number of worker threads. If kept to NULL, it resolves to available cores - 2 (min 1). Defaults to NULL.

Value

A named list with the following elements:

- walks_per_node - Integer. Number of random walks per node of the metapath's starting type. Defaults to 40L.
- walk_length - Integer. Length of each random walk. Rounded up so the walk closes on a full number of schema cycles. Defaults to 40L.
- num_workers - Integer or NULL. Number of worker threads. If kept to NULL, it resolves to available cores - 2 (min 1). Defaults to NULL.
- n_epochs - Integer. Number of training epochs. Defaults to 20L.
- n_negatives - Integer. Number of negative samples. Defaults to 5L.
- window_size - Integer. Context window size. Defaults to 2L.
- lr - Numeric. Learning rate. Defaults to 0.01.
- sample - Numeric. Subsampling threshold for frequent nodes. Defaults to 0.001.

params_node2vec	<i>Wrapper function for the node2vec parameters</i>
-----------------	---

Description

These are general node2vec parameters, designed for speed. They will default to available cores - 2 (min 1).

Usage

```
params_node2vec(
  p = 1,
  q = 1,
  walks_per_node = 40L,
  walk_length = 40L,
  batch_size = 256L,
  n_epochs = 20L,
  n_negatives = 5L,
  window_size = 2L,
  lr = 0.01,
  num_workers = NULL
)
```

Arguments

p	Numeric. Return parameter for biased random walks. Defaults to 1.0.
q	Numeric. In-out parameter for biased random walks. Defaults to 1.0.
walks_per_node	Integer. Number of random walks per node. Defaults to 40L.
walk_length	Integer. Length of each random walk. Defaults to 40L.
batch_size	Integer. Batch size for training. Defaults to 256L.

n_epochs	Integer. Number of training epochs. Defaults to 20L.
n_negatives	Integer. Number of negative samples. Defaults to 5L.
window_size	Integer. Context window size. Defaults to 2L.
lr	Numeric. Learning rate. Defaults to 0.01.
num_workers	Integer or NULL. Number of worker threads. If kept to NULL, it resolves to available cores - 2 (min 1). Defaults to NULL.

Value

A named list with the following elements:

- p - Numeric. Return parameter for biased random walks. Defaults to 1.0.
- q - Numeric. In-out parameter for biased random walks. Defaults to 1.0.
- walks_per_node - Integer. Number of random walks per node. Defaults to 40L.
- walk_length - Integer. Length of each random walk. Defaults to 40L.
- num_workers - Integer or NULL. Number of worker threads. If kept to NULL, it resolves to available cores - 2 (min 1). Defaults to NULL.
- batch_size - Integer. Batch size for training. Defaults to 256L.
- n_epochs - Integer. Number of training epochs. Defaults to 20L.
- n_negatives - Integer. Number of negative samples. Defaults to 5L.
- window_size - Integer. Context window size. Defaults to 2L.
- lr - Numeric. Learning rate. Defaults to 0.01.

plot_gw_results	<i>Plot the gene walk results</i>
-----------------	-----------------------------------

Description

Plots the GeneWalk results. Creates an interactive plotly plot to explore the results interactively.

Usage

```
plot_gw_results(object, fdr_threshold = 0.25)
```

Arguments

object	The GeneWalk class. The stats need to be available in the object.
fdr_threshold	Numeric. The FDR approach you wish to apply. GeneWalk is quite stringent, so, setting this to lower values can make sense.

Value

ggplot2 object with the plot for the gene

plot_similarities	<i>Plot similarity distributions</i>
-------------------	--------------------------------------

Description

Plots the cosine similarity distributions of the permuted embeddings alongside the actual values as density histograms.

Usage

```
plot_similarities(object)
```

Arguments

object	The GeneWalk class. Permutations and stats need to be available in the object.
--------	--

Value

A ggplot2 object with the similarity distributions.

reload_db	<i>Clean the cache and download the DB again</i>
-----------	--

Description

Clean the cache and download the DB again

Usage

```
reload_db()
```

rs_build_synthetic_genewalk	<i>Generate synthetic GeneWalk data with controlled signal structure</i>
-----------------------------	--

Description

Generate synthetic GeneWalk data with controlled signal structure

Usage

```
rs_build_synthetic_genewalk(
  n_signal_genes,
  n_noise_genes,
  n_roots,
  depth,
  branching,
  p_lateral,
  p_ppi,
  min_annotations,
  max_annotations,
  min_noise_subtrees,
  seed
)
```

Arguments

n_signal_genes	Integer. Genes annotated to a single ontology subtree.
n_noise_genes	Integer. Genes with annotations scattered across subtrees.
n_roots	Integer. Number of ontology root terms.
depth	Integer. Depth of each ontology subtree.
branching	Integer. Average branching factor per node.
p_lateral	Numeric. Probability of lateral edges within each ontology level.
p_ppi	Numeric. PPI connection probability within gene groups.
min_annotations	Integer. Minimum annotations per gene.
max_annotations	Integer. Maximum annotations per gene.
min_noise_subtrees	Integer. Minimum number of different subtrees each noise gene must span.
seed	Integer. Random seed.

Value

A list with: ontology_edges, gene_ont_edges, ppi_edges, signal_genes, noise_genes.

rs_cosine_sim	<i>Cosine similarity between two vectors</i>
---------------	--

Description

Computes cosine similarity between two numeric vectors.

Usage

```
rs_cosine_sim(a, b)
```

Arguments

a	Numeric vector.
b	Numeric vector.

Value

Cosine similarity between the two vectors.

```
rs_differential_graph_data
```

Generate a synthetic differential graph pair for testing context-aware node embeddings

Description

Produces two graphs sharing the same backbone but differing in defined regions. Community 1 is a stable negative control, identical across both graphs. Community 2 contains a hub node demoted to a peripheral node in graph 2. Two bridge nodes span communities 2 and 3 in graph 1 but are fully embedded within community 3 in graph 2. Exclusive nodes appear in only one graph.

Usage

```
rs_differential_graph_data(n_stable, n_comm2, n_comm3, n_exclusive)
```

Arguments

n_stable	Integer. Number of nodes in the stable negative-control community.
n_comm2	Integer. Number of regular nodes in community 2, excluding the hub.
n_comm3	Integer. Number of nodes in community 3, excluding bridge nodes.
n_exclusive	Integer. Number of exclusive nodes in each graph.

Value

A named list with: g1_edges (from, to), g1_nodes (node, cluster), g2_edges (from, to), g2_nodes (node, cluster), data_info (shared_nodes, is_differential, g1_only, g2_only).

rs_diffuse	<i>Run diffusion scoring on a precomputed kernel</i>
------------	--

Description

Run diffusion scoring on a precomputed kernel

Usage

```
rs_diffuse(  
  kernel,  
  scores,  
  n_bkgd,  
  n_inputs,  
  bkgd_indices,  
  method,  
  n_perm,  
  seed  
)
```

Arguments

kernel	External pointer to DiffusionKernel
scores	Numeric. The scores.
n_bkgd	Integer. Number of background nodes.
n_inputs	Integer. Number of input columns
bkgd_indices	Integer. 0-based node indices for the background
method	String. One of c("raw", "z", "mc").
n_perm	Integer. Number of permutations (MC only)
seed	Integer RNG seed (MC only)

Value

Scores per given node

rs_diffusion_kernel *Build the diffusion kernel from sparse adjacency components*

Description

Takes the CSC slots directly from R's dgCMatrix (igraph output).

Usage

```
rs_diffusion_kernel(
  i,
  p,
  x,
  n,
  kernel,
  kernel_params,
  normalised,
  strategy,
  k,
  node_names,
  verbose
)
```

Arguments

i	Integers. Row indices (0-based, from dgCMatrix@i)
p	Integers. Column pointers (from dgCMatrix@p)
x	Numeric. Values (from dgCMatrix@x)
n	Integer. Matrix dimension
kernel	String. Kernel type string. One of c("regularised_laplacian", "commute_time", "inverse_cosine", "pstep").
kernel_params	Named list. Contains the kernel-specific parameters.
normalised	Boolean. Use normalised Laplacian
strategy	String. One of "full" or "truncated". Shall the full Eigendecomposition of a truncated Eigendecomposition be applied on the data. The latter is useful for large graphs to reduce memory pressure.
k	Integer. Number of eigenvalues for truncated
node_names	String. Character vector of node names.
verbose	Boolean. Verbosity of the function.

Value

External pointer to DiffusionKernel

rs_diffusion_profiles *Generate diffusion profiles*

Description

[Experimental] Constrained personalised PageRank over a heterogeneous graph (Ruiz et al., 2021). The graph is built once and one profile per seed is computed in parallel. The seed acts as a source; all other nodes of a sink type absorb mass.

Usage

```
rs_diffusion_profiles(
  node_types,
  from,
  to,
  weights,
  type_weight_names,
  type_weight_values,
  sink_types,
  seeds,
  directed,
  diffusion_profile_params
)
```

Arguments

node_types	Character vector. Node type per node.
from	Integer vector. 1-based node indices for edge origins.
to	Integer vector. 1-based node indices for edge destinations.
weights	Optional numeric vector. Edge weights, defaults to 1.
type_weight_names	Optional character vector. Node types for type_weight_values. NULL gives the plain random walk.
type_weight_values	Optional numeric vector. Weight per node type.
sink_types	Character vector. Node types that act as sinks.
seeds	List of integer vectors. 1-based node indices; one profile per element. A set restarts uniformly over its nodes.
directed	Boolean. Treat the graph as directed.
diffusion_profile_params	Named list with alpha, max_iter and tol.

Value

Numeric matrix of n_nodes x n_seeds. Columns sum to 1.

rs_gene_walk

*Generate GeneWalk node embeddings (multiple reps)***Description**

Trains node2vec on the original GeneWalk network `n_graph` times with different random seeds, returning a list of embedding matrices. This provides the variance across embedding reps needed for the final `log_stats` aggregation.

Usage

```
rs_gene_walk(
  from,
  to,
  weights,
  gene_walk_params,
  n_graph,
  embd_dim,
  directed,
  seed,
  verbose
)
```

Arguments

<code>from</code>	Integer vector. Node indices for edge origins.
<code>to</code>	Integer vector. Node indices for edge destinations.
<code>weights</code>	Optional numeric vector. Edge weights, defaults to 1.0.
<code>gene_walk_params</code>	Named list. Training parameters (<code>p</code> , <code>q</code> , <code>walks_per_node</code> , <code>walk_length</code> , <code>num_workers</code> , <code>n_epochs</code> , <code>num_negatives</code> , <code>window_size</code> , <code>lr</code> , <code>dim</code>).
<code>n_graph</code>	Integer. Number of independent embedding reps.
<code>embd_dim</code>	Integer. Embedding dimension.
<code>directed</code>	Boolean. Treat graph as directed.
<code>seed</code>	Integer. Random seed (incremented per rep).
<code>verbose</code>	Boolean. Controls verbosity.

Value

A list of `n_graph` numeric matrices, each `n_nodes` x embedding dim.

rs_gene_walk_perm	<i>Generate null distributions from degree-matched random networks</i>
-------------------	--

Description

For each permutation, generates a random graph via the configuration model (matching the original degree distribution), trains node2vec on it, then collects cosine similarities between each node and its unique neighbours in the random graph. This matches the original Python GeneWalk procedure.

Usage

```
rs_gene_walk_perm(
    from,
    to,
    weights,
    gene_walk_params,
    n_perm,
    embd_dim,
    directed,
    seed,
    verbose
)
```

Arguments

from	Integer vector. Node indices for edge origins.
to	Integer vector. Node indices for edge destinations.
weights	Optional numeric vector. Edge weights, defaults to 1.0.
gene_walk_params	Named list. Training parameters.
n_perm	Integer. Number of null permutations.
embd_dim	Integer. Embedding dimension.
directed	Boolean. Treat graph as directed.
seed	Integer. Random seed.
verbose	Boolean. Controls verbosity.

Value

A list of `n_perm` numeric vectors of null cosine similarities.

rs_gene_walk_test	<i>Calculate GeneWalk test statistics</i>
-------------------	---

Description

Pools all null distributions into one, then for each observed embedding rep: computes p-values, per-gene FDR, and global FDR. Finally aggregates across reps with geometric mean and 95\

Matches the paper: "we repeat the above-described network representation learning and significance testing procedures of the GWN nreps_graph times and provide the mean and 95\

Usage

```
rs_gene_walk_test(
  gene_embds_list,
  pathway_embds_list,
  null_similarities,
  connected_pathways,
  verbose
)
```

Arguments

gene_embds_list	List of n_graph gene embedding matrices (n_genes x dim).
pathway_embds_list	List of n_graph pathway embedding matrices (n_pathways x dim).
null_similarities	List of n_perm numeric vectors (null cosine similarities to be pooled).
connected_pathways	List. Per-gene integer vectors of connected pathway indices (1-indexed).
verbose	Controls verbosity.

Value

A list with per-pair statistics.

rs_kernel_node_names	<i>Pull out the node names from the kernel</i>
----------------------	--

Description

Pull out the node names from the kernel

Usage

```
rs_kernel_node_names(kernel)
```

Arguments

```
kernel          External pointer to the DiffusionKernel.
```

Value

```
A string vector from R.
```

rs_metapath2vec	<i>Generate metapath2vec embeddings</i>
-----------------	---

Description

Runs metapath-constrained random walks over a heterogeneous graph and trains the skip-gram model on them (metapath2vec, or metapath2vec++ with per-type negative sampling).

Usage

```
rs_metapath2vec(  
  node_ids,  
  node_types,  
  from,  
  to,  
  weights,  
  metapath,  
  metapath_plus,  
  metapath2vec_params,  
  embd_dim,  
  directed,  
  seed,  
  verbose  
)
```

Arguments

```
node_ids      Character vector. Node identifiers.  
node_types    Character vector. Node type per node, same length as node_ids.  
from          Integer vector. 1-based indices into node_ids for edge origins.  
to            Integer vector. 1-based indices into node_ids for edge destinations.  
weights       Optional numeric vector. Edge weights.  
metapath      String. Hyphen-separated metapath closing on its starting type, e.g. "gene-pathway-gene".  
metapath_plus Boolean. Per-type negative sampling (metapath2vec++).
```

metapath2vec_params	Named list. Training parameters (walks_per_node, walk_length, num_workers, n_epochs, n_negatives, window_size, lr, sample).
embd_dim	Integer. Embedding dimension.
directed	Boolean. Treat graph as directed.
seed	Integer. Random seed.
verbose	Boolean. Controls verbosity.

Value

A list with:

- embedding - Matrix of n_nodes x embd_dim, rows in node_names order.
- node_names - Node identifiers in row order.
- node_types - Node type per row.
- visited - Logical per row. FALSE if no surviving walk touched the node, i.e. its row is the random initialisation.
- walk_stats - Named list with the walk generation statistics.

rs_node2vec

Generate a node2vec embeddings

Description

Trains node2vec on the original provided graph.

Usage

```
rs_node2vec(
  from,
  to,
  weights,
  node2vec_params,
  embd_dim,
  directed,
  seed,
  verbose
)
```

Arguments

from	Integer vector. Node indices for edge origins.
to	Integer vector. Node indices for edge destinations.
weights	Optional numeric vector. Edge weights, defaults to 1.0.

node2vec_params	Named list. Training parameters (p, q, walks_per_node, walk_length, num_workers, n_epochs, num_negatives, window_size, lr, dim).
embd_dim	Integer. Embedding dimension.
directed	Boolean. Treat graph as directed.
seed	Integer. Random seed (incremented per rep).
verbose	Boolean. Controls verbosity.

Value

An embedding of dimension n_nodes x embedding dim.

rs_node2vec_synthetic_data	<i>Generate synthetic data for node2vec</i>
----------------------------	---

Description

Generate synthetic data for node2vec

Usage

```
rs_node2vec_synthetic_data(
  test_data,
  n_nodes_per_cluster,
  n_clusters,
  p_within,
  p_between,
  seed
)
```

Arguments

test_data	String. One of c("barbell", "caveman", "stochastic_block").
n_nodes_per_cluster	Integer. Nodes per cluster.
n_clusters	Integer. Number of clusters.
p_within	Numeric. Within-cluster edge probability (0-1).
p_between	Numeric. Between-cluster edge probability (0-1).
seed	Integer. Random seed.

Value

A list with edges and nodes.

rs_procrustes_align *Orthogonal Procrustes alignment and cosine similarity scoring*

Description

Aligns embd1 onto embd2 via orthogonal Procrustes (SVD) and returns cosine similarities per node. Both matrices must have identical dimensions and matching row order, i.e. row i in embd1 corresponds to row i in embd2.

Usage

```
rs_procrustes_align(embd1, embd2)
```

Arguments

embd1	Numeric matrix. k x d embedding matrix for graph 1.
embd2	Numeric matrix. k x d embedding matrix for graph 2.

Value

Named list with aligned (k x d matrix, embd1 rotated onto embd2) and cosine_similarities (numeric vector of length k).

rs_profile_distances *Distances between the columns of two matrices*

Description

[Experimental] SIMD-accelerated distances between every column of mat_a and every column of mat_b.

Usage

```
rs_profile_distances(mat_a, mat_b, metric)
```

Arguments

mat_a	Numeric matrix. Columns are the samples.
mat_b	Numeric matrix with the same number of rows as mat_a.
metric	String. One of c("correlation", "canberra", "l1", "l2", "cosine").

Value

Numeric matrix of ncol(mat_a) x ncol(mat_b).

`synthetic_genewalk_data`*Generate synthetic GeneWalk data*

Description

Generates a synthetic network that mimics real GeneWalk input data for testing and benchmarking. The network consists of three components: a hierarchical ontology (tree structure with lateral edges), gene-to-ontology annotations, and protein-protein interactions (PPI).

Signal genes are annotated exclusively within a single ontology subtree and form PPI edges among themselves, creating coherent graph neighbourhoods. Noise genes receive the same number of annotations but are forced to span multiple subtrees, and their PPI edges are random across the noise group. Both groups are degree-matched so that significance is driven by annotation coherence rather than connectivity alone.

Usage

```
synthetic_genewalk_data(data_params = params_genewalk_data(), seed = 42L)
```

Arguments

<code>data_params</code>	Named list. Return from <code>params_genewalk_data()</code> .
<code>seed</code>	Integer. Random seed. Defaults to 42L.

Value

A list with the following items:

full_data A data.table with columns `from` and `to` containing all edges (ontology hierarchy, gene-to-pathway, and PPI).

gene_to_pathways A data.table with columns `from` (gene) and `to` (pathway) for gene-to-ontology annotation edges only.

gene_ids Character vector of all gene identifiers (signal and noise).

pathway_ids Character vector of all ontology term identifiers.

Index

* datasets

- myc_genes, [34](#)
- build_kernel, [3](#)
- calculate_drift, [4](#)
- calculate_genewalk_stats, [5](#)
- calculate_profile_distances, [6](#)
- check_degree_distribution, [7](#)
- dedup_edges, [7](#)
- differential_graph_test_data, [8](#)
- diffuse_scores, [9](#)
- DiffusionProfiles, [10](#)
- DiffusionProfiles(), [7](#), [16](#)
- DiffusionScores, [11](#)
- DiffusionScores(), [4](#)
- download_database, [11](#)
- EmbedDrift, [12](#)
- EmbedDrift(), [4](#), [15](#), [20](#), [21](#), [25](#), [32](#)
- evaluate_node2vec_test, [13](#)
- generate_initial_emb, [13](#)
- generate_initial_emb(), [15](#), [16](#)
- generate_initial_embeddings, [14](#)
- generate_permuted_emb, [15](#)
- generate_profiles, [16](#)
- generate_profiles(), [10](#), [38](#)
- GeneWalk, [17](#)
- GeneWalk(), [5](#), [7](#), [14](#), [16](#), [21](#), [23](#), [25](#), [32](#)
- GeneWalkGenerator, [18](#)
- get_aligned_embd, [20](#)
- get_db_connection, [20](#)
- get_embedding, [21](#)
- get_embeddings, [21](#)
- get_gene_info, [22](#)
- get_gene_ontology_hierarchy, [22](#)
- get_gene_ontology_info, [23](#)
- get_gene_pathway_dt, [23](#)
- get_gene_to_go, [24](#)

- get_gene_to_reactome, [24](#)
- get_graph_dt, [25](#)
- get_graph_dts, [25](#)
- get_gw_data, [26](#)
- get_gw_data_filtered, [26](#)
- get_interactions_combined, [27](#)
- get_interactions_intact, [28](#)
- get_interactions_pc, [28](#)
- get_interactions_reactome, [29](#)
- get_interactions_signor, [29](#)
- get_interactions_string, [30](#)
- get_profiles, [30](#)
- get_reactome_hierarchy, [31](#)
- get_reactome_info, [31](#)
- get_scores, [32](#)
- get_stats, [32](#)
- metapath2vec, [33](#)
- metapath2vec(), [42](#)
- myc_genes, [34](#)
- n_nodes, [35](#)
- new_data_builder, [35](#)
- node2vec, [36](#)
- node2vec(), [8](#), [33](#)
- node2vec_test_data, [37](#)
- params_diffusion_profiles, [38](#)
- params_diffusion_profiles(), [16](#)
- params_genewalk, [38](#)
- params_genewalk_data, [40](#)
- params_genewalk_data(), [58](#)
- params_kernel, [41](#)
- params_kernel(), [4](#)
- params_metapath2vec, [42](#)
- params_metapath2vec(), [33](#)
- params_node2vec, [43](#)
- params_node2vec(), [15](#)
- plot_gw_results, [44](#)
- plot_similarities, [45](#)

reload_db, [45](#)
rs_build_synthetic_genewalk, [45](#)
rs_cosine_sim, [46](#)
rs_differential_graph_data, [47](#)
rs_diffuse, [48](#)
rs_diffusion_kernel, [49](#)
rs_diffusion_profiles, [50](#)
rs_gene_walk, [51](#)
rs_gene_walk_perm, [52](#)
rs_gene_walk_test, [53](#)
rs_kernel_node_names, [53](#)
rs_metapath2vec, [54](#)
rs_node2vec, [55](#)
rs_node2vec_synthetic_data, [56](#)
rs_procrustes_align, [57](#)
rs_profile_distances, [57](#)

synthetic_genewalk_data, [58](#)