Package ‘molnet’

October 13, 2022

Type Package

Title Predicting Differential Drug Response using Multi-Omics Networks

Version 0.1.0

Description Networks provide a means to incorporate molecular interactions into reasoning, but on the omics-level, they are currently mainly used to combine genomic and proteomic information. We here present a novel network analysis pipeline that enables integrative analysis of multi-omics data including metabolomics. It allows for comparative conclusions between two different conditions, such as tumor subgroups, healthy vs. disease, or generally control vs. perturbed.
Our approach focuses on interactions and their strength instead of on node properties and includes molecules with low abundance and unknown function. We use correlation-induced networks that are reduced and combined to form heterogeneous, multi-omics molecular networks. Prior information such as metabolite-protein interactions are incorporated. A semi-local, path-based integration step denoises the network and ensures integrative conclusions. As case studies, we investigate differential drug response in breast cancer tumor datasets providing proteomics, transcriptomics, phospho-proteomics and metabolomics data and contrasting patients with different estrogen receptor status.
Our proposed pipeline leverages multi-omics data for differential predictions, e.g. on drug response, and includes prior information on interactions.
The case study presented in the vignette uses data published by Krug (2020) <doi:10.1016/j.cell.2020.10.036>. The package license applies only to the software and explicitly not to the included data.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.1.1

VignetteBuilder knitr

Imports igraph, dplyr, stringr, WGCNA, Rfast, readr, tibble, tidyr, magrittr, rlang

Suggests rmarkdown, knitr

Depends R (>= 3.5.0)

NeedsCompilation no

Author Katharina Baum [cre] (<https://orcid.org/0000-0001-7256-0566>),
Julian Hugo [aut] (<https://orcid.org/0000-0003-3355-1071>),
Spoorthi Kashyap [aut],
Nataniel Müller [aut] (<https://orcid.org/0000-0002-0275-3992>),
Justus Zeinert [aut] (<https://orcid.org/0000-0003-3918-0507>)

Maintainer Katharina Baum <katharina.baum@hpi.de>

Repository CRAN

Date/Publication 2021-08-06 08:30:02 UTC

R topics documented:

`calculate_interaction_score`
Calls a python script to calculate interaction score for combined graphs

---

Description

(INTERNAL) The interaction score is computed and replaces the edge weight. This function expects the combined graphs for both groups along with their corresponding drug target and node lists to be present at ‘loading_path‘. Graphs and drug targets should be weighted edge lists in tsv format. Node files should contain one node id per line. The script for calculating the interaction score is called with ‘python_executable‘. An alternate script can be specified with ‘script_path‘. The score for an edge is computed as the sum of the average product of weights along all simple paths of length l (over all path lengths up to ‘max_path_length‘) between the source and target node of the edge.

Usage

```
calculate_interaction_score(
  max_path_length,
  total_edges,
  loading_path,
  python_executable = "python3",
  script_path = NULL,
  int_score_mode = "auto"
)
```

Arguments

`max_path_length`	The maximum length of simple paths to consider when computing the interaction score
`total_edges`	vector with total edges in each group
`loading_path`	Directory to use for writing intermediate data when passing input and output between Python and R
`python_executable`	Python command or path to Python executable to use
`script_path`	Path to the interaction score Python script. Set NULL to use package internal script (default).
`int_score_mode`	One of ‘auto‘, ‘sequential‘ or ‘ray‘. Whether to compute interaction score in parallel using the Ray python library or sequentially. When ‘auto‘ it depends on the graph sizes.

Value

Does not return anything, instead calls Python script which outputs .gml files

---

`check_connection` *Checks connection*

Description

(INTERNAL) Checks if the data given to create an inter-layer connection is valid and has the right input format

Usage

```
check_connection(connection)
```

Arguments

`connection`	Connection to check. Created by `make_connection`

Value

Character string vector containing error messages.

Examples

```
con = make_connection("mrna", "protein", connect_on="gene_name")
return_errors(check_connection(con))
```

---

check_drug_target *Check drug target interaction data*

Description

(INTERNAL) Checks if the data used to define interaction between drugs and targets is valid and formatted correctly.

Usage

```
check_drug_target(drug_target_interaction)
```

Arguments

`drug_target_interaction`	A named list of the drug interaction data. Created by `make_drug_target`

Value

Character string vector containing error messages.

Examples

```
data(drug_gene_interactions)
drug_target_interaction <- make_drug_target(target_molecules='protein',
interaction_table=drug_gene_interactions,
match_on='gene_name')
return_errors(check_drug_target(drug_target_interaction))
```

`check_drug_targets_in_layers` *Check drug target and layer data*

Description

(INTERNAL) Checks if the parameters supplied in 'drug_target_interaction' makes sense in the context of the defined layers.

Usage

```
check_drug_targets_in_layers(drug_target_interaction, layers)
```

Arguments

`drug_target_interaction`	A named list of the drug interaction data. Created by `make_drug_target`
`layers`	List of layers to check. Individual layers are created by `make_layer` and need to be wrapped in a list.

Value

Character string vector containing error messages.

Examples

```
data(layers_example)
layers <- layers_example
data(drug_gene_interactions)
drug_target_interaction <- make_drug_target(target_molecules='protein',
interaction_table=drug_gene_interactions,
match_on='gene_name')
return_errors(check_drug_targets_in_layers(drug_target_interaction, layers))
```

`check_input` *Check pipeline input data for required format*

Description

Checks if input data is valid and formatted correctly. This function is a wrapper for other check functions to be executed as first step of the molnet pipeline.

Usage

```
check_input(layers, inter_layer_connections, drug_target_interaction)
```

Arguments

`layers`	List of layers to check. Individual layers were created by `make_layer` and need to be wrapped in a list.
`inter_layer_connections`	A list containing connections between layers. Each connection was created by `make_connection` and wrapped in a list.
`drug_target_interaction`	A named list of the drug interaction data. Created by `make_drug_target`

Value

Character string vector containing error messages.

`check_layer` *Check layer input*

Description

(INTERNAL) Checks if the data used to create a network layer is valid and has the right format

Usage

```
check_layer(layer)
```

Arguments

`layer`	layer to check. Created by `make_layer`

Value

Character string vector containing error messages.

Examples

```
data(metabolite_data)
metabolite_layer = make_layer(name="metabolite",
metabolite_data$group1$data,
metabolite_data$group2$data,
metabolite_data$group1$identifiers,
metabolite_data$group2$identifiers)
return_errors(check_layer(metabolite_layer))
```

check_sensible_connections

Check connection and layer data

Description

(INTERNAL) Checks if the connection defined in 'connection' makes sense in context of the defined layers.

Usage

```
check_sensible_connections(connection, layers)
```

Arguments

`connection`	Connection to check. Created by `make_connection`
`layers`	List of layers to check. Individual layers are created by `make_layer` and need to be wrapped in a list.

Value

Character string vector containing error messages.

Examples

```
data(mrna_data)
mrna_layer = make_layer(name="mrna",
mrna_data$group1$data, mrna_data$group2$data,
mrna_data$group1$identifiers,
mrna_data$group2$identifiers)
data(protein_data)
protein_layer = make_layer(name="protein",
protein_data$group1$data,
protein_data$group2$data,
protein_data$group1$identifiers,
protein_data$group2$identifiers)
con = make_connection("mrna", "protein", connect_on="gene_name")
return_errors(check_sensible_connections(con, layers=list(mrna_layer, protein_layer)))
```

`chunk` *Create chunks from a vector for parallel computing*

Description

(INTERNAL)

Usage

```
chunk(x, n)
```

Arguments

`x`	vector
`n`	length of chunks

Value

a list of chunks of length n

Source

https://stackoverflow.com/questions/3318333/split-a-vector-into-chunks

`chunk_2gether` *Create chunks from two vectors for parallel computing*

Description

(INTERNAL)

Usage

```
chunk_2gether(x, y, n)
```

Arguments

`x, y`	vectors
`n`	length of chunks

Value

A list of lists. Each second level list contains a list of chunks of length n of each input vector.

Source

modified from: https://stackoverflow.com/questions/3318333/split-a-vector-into-chunks `combined_graphs_example` *Combined graphs*

Description

Exemplary intermediate pipeline output: Combined graphs example data built by `generate_combined_graphs`. Graphs were created with default settings of `molnet_settings`. A subset of the original data by Krug et al., 2020 and randomly sampled metabolite data was used to generate graphs `layers_example`. They were created from data stratified by estrogen receptor (ER) status: group1 contains data of ER+ patients and group2 of ER- patients.

Usage

```
combined_graphs_example
```

Format

A named list with 2 items.

graphs A named list with two groups.

- **group1** Graph associated with group1
- **group2** Graph associated with group2

annotations A named list containing data frames of mappings of assigned node IDs to the user-provided component identifiers for nodes in group1 or group2 and all nodes

- **group1**
- **group2**
- **all**

Source

Krug, Karsten et al. “Proteogenomic Landscape of Breast Cancer Tumorigenesis and Targeted Therapy.” Cell vol. 183,5 (2020): 1436-1456.e31. doi:10.1016/j.cell.2020.10.036

`combine_graphs` *Combining graphs by adding inter-layer edges*

Description

Creates the union of all graphs and adds the inter-layer edges.

Usage

```
combine_graphs(graphs, inter_layer_edgelists)
```

Arguments

`graphs`	List of iGraph objects
`inter_layer_edgelists`	List of data frames containing inter-layer edges

Value

iGraph object which is the union of the input graphs with isolated nodes removed.

`corPvalueStudentParallel` *Compute p-values for upper triangle of correlation matrix in parallel*

Description

(INTERNAL)

Usage

```
corPvalueStudentParallel(correlation, n_samples, chunk_size)
```

Arguments

`correlation`	matrix of correlation values
`n_samples`	matrix of number of samples used in computation of each correlation value
`chunk_size`	smallest unit of work in parallel computation (number of p-values to compute)

Value

vector of p-values for upper triangle

`create_unique_layer_node_ids` *Assigns node IDs to the biological identifiers across a graph layer*

Description

(INTERNAL) This function takes two data frames of (biological) identifiers of nodes. Each data frame corresponds to the identifiers of the components contained in the single-layer network of a sample group. This function outputs the same data frames, with an added column (‘node_id‘) that contains node IDs which can later be used as ‘name‘ parameter for an iGraph graph. Node IDs begin with the defined ‘prefix‘ and an underscore. If a molecule is present in both groups, the node ID will be the same across the whole layer, allowing to easily combine the graphs of both groups in ‘differential_score()‘ to calculate differential scores of identical nodes in both sample groups. The function is used by the high-level wrapper generate_individual_graphs to create annotations, which uniquely define nodes across the network layer.

Usage

```
create_unique_layer_node_ids(identifiers1, identifiers2, layer_name)
```

Arguments

`identifiers1, identifiers2`	Data frames containing the biological identifiers of each group of the same network layer.
`layer_name`	Name of layer node ids are created for

Value

Returns an named list. Elements ‘identifiers1‘ and ‘identifiers2‘ contain the input data frames with an additional column ‘node_id‘. ‘all‘ contains all unique node IDs assigned across the network layer.

`determine_drug_targets` *Determine drug target nodes in network*

Description

Finds node IDs of network nodes in ‘graphs‘ that are targeted by a drug in ‘drug_target_interaction‘. Returns list of node ids and list of adjacent edges.

Usage

```
determine_drug_targets(graphs, annotations, drug_target_interaction, settings)
```

Arguments

`graphs`	A named list with elements ‘group1‘ and ‘group2‘ containing the combined graphs of each group as iGraph object.
`annotations`	List of data frames that map node IDs to identifiers. Contains ‘all‘ (unique identifiers across the whole data) and ‘group1‘ and ‘group2‘ containing identifiers specific to the strata.
`drug_target_interaction`	Named list specifying drug target interactions for drug response score computation
`settings`	A named list containing pipeline settings

Value

A named list with elements ‘drug_targets‘ and ‘drug_target_edge_list‘. * ‘targets‘ is a named list with elements ‘target_nodes‘ and ‘drugs_to_target_nodes‘. ‘target_nodes‘ is a data frame with column ‘node_id‘ (unique node IDs in the iGraph object targeted by drugs) and columns ‘group1‘ and ‘group2‘ (boolean values specifying whether the node is contained in the combined graph of the group). Element ‘drugs_to_target_nodes‘ contains a named list mapping drug names to a vector of their target node IDs. * ‘drug_target_edge_list‘ contains elements ‘group1‘ and ‘group2‘ containing each a list of edges adjacent to drug target nodes.

Examples

```
data(drug_gene_interactions)
data(combined_graphs_example)
combined_graphs <- combined_graphs_example
settings <- molnet_settings()
drug_target_interaction <- make_drug_target(target_molecules='protein',
interaction_table=drug_gene_interactions,
match_on='gene_name')
drug_targets <- determine_drug_targets(combined_graphs[["graphs"]],
combined_graphs[["annotations"]],
drug_target_interaction,
settings)
```

`differential_score` *The absolute difference of interaction score of two groups*

Description

Computes the absolute difference of interaction score between two groups. Returns a single graph with the differential score as only edge attribute. The interaction score is computed by `interaction_score`.

Usage

```
differential_score(interaction_score_graphs, score_name = "weight")
```

Arguments

`interaction_score_graphs`	Named list with elements ‘group1‘ and ‘group2‘ containing iGraph objects with score as edge attribute. Output of `interaction_score`.
`score_name`	Character string specifying the name of the edge attribute (default: ‘weight‘).

Value

iGraph object with ‘differential_score‘ as only edge attribute

Examples

```
data(interaction_score_graphs_example)
interaction_score_graphs <- interaction_score_graphs_example
differential_score_graph <- differential_score(interaction_score_graphs, score_name = "weight")
```

---

differential_score_graph_example
Differential graph

Description

Exemplary intermediate pipeline output: Differential score graph example data built by `differential_score`. Contains one graph carrying the differential interaction score as weight. This was computed using a subset of the data published by Krug et al., 2020 and randomly sampled metabolite data `layers_example`.

Usage

```
differential_score_graph_example
```

Format

An iGraph graph object.

Source

Krug, Karsten et al. “Proteogenomic Landscape of Breast Cancer Tumorigenesis and Targeted Therapy.” Cell vol. 183,5 (2020): 1436-1456.e31. doi:10.1016/j.cell.2020.10.036

---

drug_gene_interactions
Drug-gene interactions

Description

Data frame providing interactions of drugs with genes. The data was downloaded from The Drug Gene Interaction Database.

Usage

```
drug_gene_interactions
```

Format

A data frame with 4 columns.

gene_name Gene names of targeted protein-coding genes.

ncbi_id NCBI IDs of targeted protein-coding genes.

drug_name Drug-names with known interactions.

drug_chembl_id ChEMBL ID of drugs.

Source

The Drug Gene Interaction Database: `https://www.dgidb.org/`

ChEMBL IDs: `https://www.ebi.ac.uk/chembl`

`drug_response_score_example` *Drug response score*

Description

Exemplary final pipeline output: Drug response score data frame. This contains drugs and the calculated differential drug response score. The score was calculated by `get_drug_response_score`.

Usage

```
drug_response_score_example
```

Format

Data frame with two columns

drug_name Names of drugs

drug_response_score Associated differential drug response scores

Details

The original data used to compute this object was data published by Krug et al., 2020 and randomly sampled metabolite data `layers_example`.. Drug-gene interactions to calculate this output were used from The Drug Gene Interaction Database.

Source

Krug, Karsten et al. "Proteogenomic Landscape of Breast Cancer Tumorigenesis and Targeted Therapy." Cell vol. 183,5 (2020): 1436-1456.e31. doi:10.1016/j.cell.2020.10.036

The Drug Gene Interaction Database: `https://www.dgidb.org/`

`drug_targets_example` *Drug target nodes in combined network*

Description

Exemplary intermediate pipeline output: Drug targets detected in the combined graphs. A named list with elements ‘drug_targets‘ and ‘edgelist‘. This was created from `determine_drug_targets` using the default settings given by `molnet_settings`. Drug-gene interactions to calculate this output were used from The Drug Gene Interaction Database.

Usage

```
drug_targets_example
```

Format

A named list with 2 items.

targets A named list

- **target_nodes** data frame with column ‘node_id‘ (unique node IDs in the graph targeted by drugs) and columns ‘group1‘ and ‘group2‘ (boolean values specifying whether the node is contained in the combined graph of the group)
- **drugs_to_target_nodes** Element ‘drugs_to_target_nodes‘ contains a named list mapping drug names to a vector of their target node IDs.

edgelist Contains elements ‘group1‘ and ‘group2‘ containing each a data frame of edges adjacent to drug target nodes each. Each edgelist data frame contains columns ‘from‘, ‘to‘ and ‘weight‘.

Source

The Drug Gene Interaction Database: `https://www.dgidb.org/`

`drug_target_interaction_example` *Drug target interaction example data*

Description

Drug target interaction example data

Usage

```
drug_target_interaction_example
```

Format

A named list with 3 items.

target_molecules Name of layer containing the drug targets. This name has to match the corresponding named item in the list of layers supplied to `start_pipeline`.

interaction_table Table giving drug-gene-interactions.

- **gene_name** Gene names of targeted protein-coding genes.
- **ncbi_id** NCBI IDs of targeted protein-coding genes.
- **drug_name** Drug-names with known interactions.
- **drug_chembl_id** ChEMBL ID of drugs.

match_on Column name of the data frame supplied in ‘interaction_table‘ that is used for matching drugs and target nodes in the graph (e.g. ‘ncbi_id‘).

Source

Terunuma, Atsushi et al. “MYC-driven accumulation of 2-hydroxyglutarate is associated with breast cancer prognosis.” The Journal of clinical investigation vol. 124,1 (2014): 398-412. doi:10.1172/JCI71180

`https://www.metabolon.com`

Pubchem IDs: `https://pubchem.ncbi.nlm.nih.gov`

MetaboAnalyst: `https://www.metaboanalyst.ca/faces/upload/ConvertView.xhtml`

`find_targets` *Filter drug target nodes*

Description

(INTERNAL) Based on the supplied target molecules, interaction table, graph and annotation this function returns a data frame containing nodes in the network targeted by a drug and a list containing the drug names as names and a vector of node IDs as keys.

Usage

```
find_targets(graphs, target_molecules, interaction_table, annotation, on)
```

Arguments

`graphs`	List of two iGraph graph objects (one for each group)
`target_molecules`	Character string. Identifies the type of the target molecules (e.g., ‘protein‘). The string must be contained in the ‘type‘ column of the annotation data frame.
`interaction_table`	Data frame. Specifying the interaction of drugs and target molecules. Must contain a column ‘drug_name‘ containing drug names/identifiers and a column named like the character string given in the ‘on‘ argument, which must be an identifier for the targeted molecule.

`annotation`	Data frame. Contains the annotation for all the nodes contained in the combined network. Must contain a column 'node_id' (vertex IDs in iGraph graph object) and a column named like the character string given in the 'on' argument, which must be an identifier for the targeted molecule.
`on`	Character string. Defines the ID that is used to match drugs to their targets. Both supplied data frames ('annotation' and 'interaction_table') must contain a column named like this character string.

Value

A named list. Element 'target_nodes' is a data frame with column 'node_id' (unique node IDs in the iGraph graph object that are targeted by drugs) and columns 'group1' and 'group2' (boolean values specifying whether the node is contained in the combined graph of the group). Element 'drugs_to_target_nodes' contains a named list: elements are 'drug_names' and contain a vector of node IDs that are their specific targets.

`generate_combined_graphs`

Combines individual layers to a single graph

Description

Individual graphs created by `generate_individual_graphs` are combined to a single graph per group according to 'inter_layer_connections'. Returns a list of combined graphs along with their annotations.

Usage

```
generate_combined_graphs(
  graphs,
  annotations,
  inter_layer_connections,
  settings
)
```

Arguments

`graphs`	A named list (elements 'group1' and 'group2'). Each element contains a list of iGraph objects (output of `generate_individual_graphs`).
`annotations`	A named list (elements 'group1' and 'group2'). Each element contains a list of data frames mapping each node IDs to identifiers (output of `generate_individual_graphs`).
`inter_layer_connections`	Named list with specified inter-layer connections. Names are layer names and elements are connections (make_connection).
`settings`	A named list containing pipeline settings

Value

A named list (elements ‘graphs‘ and ‘annotations‘ and sub-elements ‘$group1‘ and ‘$group2‘). Contains the igraph objects of the combined network and their annotations for both groups.

Examples

```
data(individual_graphs_example)
individual_graphs <- individual_graphs_example
inter_layer_connections <- list(molnet::make_connection(from="mrna",
to="protein", connect_on="gene_name", weight=1))
settings <- molnet::molnet_settings() # defaults
combined_graphs <- molnet::generate_combined_graphs(individual_graphs$graphs,
                                                    individual_graphs$annotations,
                                                    inter_layer_connections,
                                                    settings)
```

`generate_individual_graphs`
Builds graphs from specified network layers

Description

Constructs and returns two graphs for each network layer, where nodes correspond to the rows in the measurement data. Graphs are initially complete and edges are weighted by correlation of measurements across columns. The number of edges is then reduced by either a threshold on the p-value of the correlation or a minimum scale-free fit index. Each node is mapped to the biological identifiers given in the layer and the mapping table is returned as ‘annotations‘.

Usage

```
generate_individual_graphs(layers, settings)
```

Arguments

`layers`	Named list with different network layers containing data and identifiers for both groups (generated from `make_layer`)
`settings`	A named list containing pipeline settings

Value

A nested named list with first-level elements ‘graphs‘ and ‘annotations‘. The second level elements are ‘group1‘ and ‘group2‘. These contain a list of iGraph objects (‘graphs‘) and data frames (‘annotations‘) mapping the graph node IDs to biological identifiers.

Examples

```
data(layers_example)
layers <- layers_example
settings <- molnet::molnet_settings(handling_missing_data="pairwise.complete.obs")
individual_graphs <- molnet::generate_individual_graphs(layers, settings)
molnet::graph_metrics(individual_graphs$graphs$group1$mrna)
molnet::graph_metrics(individual_graphs$graphs$group2$mrna)
```

`generate_reduced_graph` *Generate a reduced iGraph*

Description

(INTERNAL) A wrapper functions that calls the functions to generate a network from raw data and reduce the network by a given method. Graph generation is using `graph.adjacency` internally. Methods implemented are network_reduction_by_p_value (reduction by statistical significance of correlation) and network_reduction_by_pickHardThreshold (using WGCNA function pickHardThreshold.fromSimilarity that finds a suitable cutoff value to get a scale-free network). If no method is given, no reduction will be performed. When using the reduction method ‘p_value‘ the user can specify an alpha significance value and a method for p-value adjustment. When using the reduction by ‘pickHardthreshold‘ a R-Squared Cutoff can be specified and a cut vector can be supplied. The adjacency matrix of correlations is computed using cor. The handling of missing data can be specified. Both the adjacency of correlations and the graph object can be saved optionally.

Usage

```
generate_reduced_graph(
  measurement_data,
  identifiers,
  correlation_method = "spearman",
  reduction_method = "p_value",
  save_correlation_filename = NULL,
  handling_missing_data = "all.obs",
  p_value_adjustment_method = "BH",
  reduction_alpha = 0.05,
  r_squared_cutoff = 0.85,
  cut_vector = seq(0.2, 0.8, by = 0.05),
  print_graph_info = FALSE,
  n_threads = parallel::detectCores() - 1,
  parallel_chunk_size = 10^6
)
```

Arguments

`measurement_data`	Data frame containing raw data (e.g. mRNA expression data, protein abundance, etc.). Analyzed components (e.g. genes) in rows, samples (e.g. patients) in columns.
`identifiers`	Data frame containing biological identifiers and the corresponding node ID created in create_unique_layer_node_ids. The column containing node IDs has to be named ‘node_id‘.
`correlation_method`	A character string specifying the method to be used for correlation calculation by cor. Can be any of "spearman", "pearson" or "kendall".
`reduction_method`	A character string specifying the method to be used for network reduction. ‘p_value‘ for hard thresholding based on the statistical significance of the computed correlation. ‘pickHardThreshold‘ for a cutoff based on the scale-freeness criterion (calls ‘WGCNA::pickHardThreshold‘).
`save_correlation_filename`	(optional) Set a name for saving the matrix of correlations. Will be saved as .rds file.
`handling_missing_data`	A character string specifying the handling of missing data. Use "all.obs" (default) or "pairwise.complete.obs". Argument is passed to ‘WGCNA::cor()‘.
`p_value_adjustment_method`	String of the correction method applied to p-values. Passed to p.adjust.
`reduction_alpha`	A number indicating the alpha value applied for thresholding.
`r_squared_cutoff`	A number indicating the desired minimum scale free topology fitting index R^2.
`cut_vector`	A vector of hard threshold cuts for which the scale free topology fit indices are to be calculated.
`print_graph_info`	A boolean value specifying if a summary of the reduced graph should be printed.
`n_threads`	Number of threads for parallel computation of p-values during p-value reduction.
`parallel_chunk_size`	Number of p-values in smallest work unit when computing in parallel during network reduction with method ‘p_value‘.

Value

iGraph graph object of the reduced network.

`get_drug_response_score` *Calculate drug response score*

Description

This function takes the differential graph (generated in `differential_score`), the a drug targets object (containing target node names and drugs and their targets; generated in `determine_drug_targets`) and the supplied drug-target interaction table (formatted in `make_drug_target`) to calculate the differential drug response score. The score is the median of all differential scores of the edges adjacent to all drug target nodes of a particular drug.

Usage

```
get_drug_response_score(differential_graph, drug_targets, interaction_table)
```

Arguments

`differential_graph`	iGraph graph object containing differential scores for all edges. Output of `differential_score`
`drug_targets`	Named list containing two elements (‘target_nodes‘ and ‘drugs_to_target_nodes‘). Output of `determine_drug_targets`. ‘target_nodes‘ is a vector containing network node names of the nodes that are targeted by the available drugs. ‘drugs_to_target_nodes‘ is a dictionary-like list that maps drugs to the nodes that they target.
`interaction_table`	Data frame. Element ‘interaction_table‘ of ‘drug_target_interaction‘ created by `make_drug_target`. Contains at least two columns: ‘drug_name‘ containing names of drugs and a column named with an identifier present in the targeted layer.

Value

Data frame containing drug name and associated differential drug response score

Examples

```
data(drug_gene_interactions)
drug_target_interaction <- make_drug_target(target_molecules='protein',
interaction_table=drug_gene_interactions, match_on='gene_name')


data(drug_targets_example)
data(differential_score_graph_example)
drug_response_score <- get_drug_response_score(differential_score_graph_example,
drug_targets_example[["targets"]], drug_target_interaction$interaction_table)
```

`get_layer` *[INTERNAL] Fetch layer by name from layer object*

Description

[INTERNAL] Fetch layer by name from layer object

Usage

```
get_layer(name, layers)
```

Arguments

`name`	The layer to fetch
`layers`	a layers object `layers_example`

Value

Returns the layer along with layer names

`get_layer_setting` *Get layer settings*

Description

Returns specified setting for a specific network layer.

Usage

```
get_layer_setting(layer, settings, setting_name)
```

Arguments

`layer`	A network layer created by `make_layer`
`settings`	Named list of settings created by `molnet_settings`
`setting_name`	String indicating the setting to return.

Value

Setting value(s) for this layer

graph_metrics *Analyses metrics of an iGraph object*

Description

This helper function prints or returns multiple metrics of arbitrary iGraph graph object.

Usage

```
graph_metrics(graph, verbose = TRUE, return = FALSE)
```

Arguments

graph	iGraph object to analyze.
verbose	If TRUE graph information is printed.
return	If TRUE graph information is returned from function.

Value

Named list of metrics including vertex count, edge count, number of components, size of largest component and the relative frequency of zero degree vertices.

Examples

```
adj_mat <- matrix(rnorm(36),nrow=6)
graph <- igraph::graph_from_adjacency_matrix(adj_mat)
graph_metrics(graph, verbose = TRUE, return = FALSE)
```

individual_graphs_example *Individual graphs*

Description

Exemplary intermediate pipeline output: Individual graphs example data built by `generate_individual_graphs`. Graphs were created by correlation computation and reduced by the 'p_value' reduction method (default settings of `molnet_settings`. A subset of the original data by Krug et al., 2020 and randomly sampled metabolite data (`layers_example`) was used to generate graphs. They were created from data tratified by estrogen receptor (ER) status: group1 contains data of ER+ patients and group2 of ER- patients.

Usage

```
individual_graphs_example
```

Format

A named list with 2 items.

graphs A named list with two groups.

- **group1** Graphs associated with group1
  - **mrna**
  - **protein**
  - **phosphoprotein**
  - **metabolite**
- **group2** same structure as above

annotations A named list containing data frames of mappings of assigned node IDs to the user-provided component identifiers for nodes in group1 or group2 and all nodes

- **group1**
- **group2**
- **all**

Source

Krug, Karsten et al. "Proteogenomic Landscape of Breast Cancer Tumorigenesis and Targeted Therapy." Cell vol. 183,5 (2020): 1436-1456.e31. doi:10.1016/j.cell.2020.10.036

`install_python_dependencies`

Installs python dependencies needed for interaction score computation

---

Description

Uses specified pip or conda executable (default: pip3) to install all required python modules. When using conda, the currently active environment is used. Commands run are 'pip install -r requirements' or 'conda install –file requirements'. Installs the following requirements: - numpy - tqdm - python-igraph - ray

Usage

```
install_python_dependencies(package_manager = "pip3")
```

Arguments

`package_manager`
The package manager command or path to use (default: pip3)

`interaction_score` *Computes interaction score for combined graphs*

Description

Writes the input data (combined graphs for both groups in gml format and lists of edges adjacent to drug targets for both groups) to files and calls a python script for calculating the score. Output files written by the python script are two graphs in gml format containing the interaction score as weight. These are loaded and returned in a named list. ATTENTION: Data exchange via files is mandatory and takes a long for large data. Interaction score computation is expensive and slow because it involves finding all simple paths up to a certain length between source and target node of the drug target edges. Don't set 'max_path_length' in settings to a large value and only consider this step if your graphs have up to approximately 2 million edges. Computation is initiated by `calculate_interaction_score`. The python script is parallelized using Ray. Use the setting 'int_score_mode' to force sequential or parallel computation. Refer to the Ray documentation if you encounter problems with running the python script in parallel. DISCLAIMER: Depending on the operating system Python comes pre-installed or has to be installed manually. Please pay attention to the version and the executable used (python/python3 or homebrew python). You can use the 'python_executable' setting to specify the command or path.

Usage

```
interaction_score(graphs, drug_target_edgelists, settings)
```

Arguments

`graphs`	A named list (elements 'group1' and 'group2'). Each element contains the combined graph for its group.
`drug_target_edgelists`	A named list (elements 'group1' and 'group2'). Each element contains the list of edges adjacent to drug targets as a data frame (columns 'from', 'to' and 'weight')
`settings`	A named list containing pipeline settings

Value

A named list (elements 'group1' and 'group2'). Each element contains an iGraph object containing the interaction score as weight.

Examples

```
data(combined_graphs_example)
data(drug_targets_example)
settings <- molnet_settings()

interaction_score_graphs <- interaction_score(combined_graphs_example[["graphs"]],
drug_target_edgelists=drug_targets_example[["edgelist"]],
settings=settings)
```

`interaction_score_graphs_example`
Interaction score graphs

Description

Exemplary intermediate pipeline output: Interaction score graphs example data built by `interaction_score`. A named list (elements ‘group1‘ and ‘group2‘). Each element contains an iGraph object containing the interaction score as weight. This was computed using a subset of the data published by Krug et al., 2020 and randomly sampled metabolite data `layers_example`.

Usage

```
interaction_score_graphs_example
```

Format

A named list with 2 items.

group1 iGraph graph object containing the interaction score as weight for group1.

group2

Source

Krug, Karsten et al. “Proteogenomic Landscape of Breast Cancer Tumorigenesis and Targeted Therapy.” Cell vol. 183,5 (2020): 1436-1456.e31. doi:10.1016/j.cell.2020.10.036

`interaction_score_graphs_vignette`
Interaction score graphs for vignette

Description

Exemplary intermediate pipeline output used in the vignette.

Usage

```
interaction_score_graphs_vignette
```

Format

A named list with 2 items.

group1 iGraph graph object containing the interaction score as weight for group1.

group2

Source

Krug, Karsten et al. “Proteogenomic Landscape of Breast Cancer Tumorigenesis and Targeted Therapy.” Cell vol. 183,5 (2020): 1436-1456.e31. doi:10.1016/j.cell.2020.10.036

`inter_layer_edgelist_by_id`
Interlayer conntections by identifiers

Description

(INTERNAL) Returns an edge list defining the connections between two layers of the network.

Usage

```
inter_layer_edgelist_by_id(annotation_1, annotation_2, connection, weight = 1)
```

Arguments

`annotation_1, annotation_2`	Data frames: Annotation tables specifying the identifiers of the nodes of a network
`connection`	String of identifier to connect on
`weight`	Integer or vector specifying the weight of the inter-layer connections.

Value

data.frame with columns from, to and weight

inter_layer_edgelist_by_table *Interaction table to iGraph graph object*

Description

(INTERNAL) Returns an edge list defining the connections between two layers of the network based on an interaction table supplied by the user.

Usage

```
inter_layer_edgelist_by_table(
  annotation_1,
  annotation_2,
  interaction_table,
  weight_column
)
```

Arguments

`annotation_1, annotation_2`	Data frames: Annotation tables specifying the identifiers of the nodes of a network
`interaction_table`	Table specifying the interaction / connections between the two layers
`weight_column`	Name of the column in 'interaction_table' giving the weight of the inter-layer edges.

Value

data.frame with columns from, to and weight

layers_example *Formatted layers object*

Description

Exemplary intermediate pipeline output containing a correctly formatted layers list.

Usage

```
layers_example
```

Format

A list with 4 items. Each layer list contains 2 groups and a 'name' element. Each group contains 'data' and 'identifiers'. The structure for one individual layer:

group1 Data associated with group1

- **data** Raw data. Components (e.g. genes) in columns, samples in rows
- **identifiers** Data frame containing one column per ID

group2 Data associated with group2

- **data** see above
- **identifiers** see above

name Name of the layer

Details

List containing four layer items created by `make_layer`. Each layer contains 'data' and 'identifiers' stratified by group and a 'name' element giving the layer name. The data contained in this example refers to mRNA, protein, phosphoprotein and metabolite layers. The data on mRNA, protein and phosphoproteins in taken from Krug et al., 2020 containing data from the Clinical Proteomic Tumor Analysis Consortium (CPTAC). The metabolite data was sampled randomly to generate distributions similar to those reported (e.g., in Terunuma et al., 2014).

Source

Terunuma, Atsushi et al. "MYC-driven accumulation of 2-hydroxyglutarate is associated with breast cancer prognosis." The Journal of clinical investigation vol. 124,1 (2014): 398-412. doi:10.1172/JCI71180

Krug, Karsten et al. "Proteogenomic Landscape of Breast Cancer Tumorigenesis and Targeted Therapy." Cell vol. 183,5 (2020): 1436-1456.e31. doi:10.1016/j.cell.2020.10.036

`load_interaction_score_output` *Loads output of python script for interaction score calculation*

Description

(INTERNAL) Loads data generated by `calculate_interaction_score`. Output files are graphs in gml format for both groups.

Usage

```
load_interaction_score_output(loading_path)
```

Arguments

`loading_path`	Directory to load from

Value

A named list (elements ‘group1‘ and ‘group2‘). Each element contains an iGraph object containing the interaction score as edge attribute.

`make_connection` *Specify connection between two individual layers*

Description

Helper function to transform input data to a required pipeline input format. This helper function creates a list that specifies the connection between two layers.

Usage

```
make_connection(from, to, connect_on, weight = 1, group = "both")
```

Arguments

`from`	Character string referring to the name of the layer **from** which the connection should be established
`to`	Character string referring to the name of the layer **to** which the connection should be established
`connect_on`	Specifies how the two layers should be connected. This can be based on a mutual ID or a table specifying interactions: * __Mutual ID__: Character string specifying the name of an identifier that is present in both layers (e.g., ‘NCBI ID‘ to connect proteins and mRNA). * __Interaction table__: A table mapping two identifiers of two layers. The columns have exactly the same names as the identifiers of the layers. Has to contain an additional column specifying the weight between two components/nodes (see ‘weight‘ argument)
`weight`	Specifies the edge weight between the layers. This can be supplied as a number applied to every connection or a column of the interaction table: * __Fixed weight__: number specifying the weight of every connection between the layers. * __Based on interaction table__: Character string specifying the name of a column in the table passed as the ‘by‘ parameter which is used as edge weight.
`group`	Group for which to apply the connection. One of ‘both‘, ‘1‘ or ‘2‘.

Details

The connection can be based on IDs present in the identifiers of both layer or an interaction table containing mapping the connections and edge weights. Additionally, the supplied input is checked. Allows easy conversion of raw data into the structure accepted by `start_pipeline`.

__IMPORTANT:__ if a connection is established based on `id` this ID has to be present in the identifiers of both layers, have to be named identically and IDs have to be formatted identically as these are matched by an inner join operation (refer to `make_layer`).

Value

A named list (i.e., an inter-layer connection), that can be supplied to `start_pipeline`.

Examples

```
data(metabolite_protein_interaction)
inter_layer_connections = list(
make_connection(from = 'mrna', to = 'protein', connect_on = 'gene_name'),
make_connection(from = 'protein', to = 'phosphoprotein', connect_on = 'gene_name'),
make_connection(from = 'protein', to = 'metabolite',
connect_on = metabolite_protein_interaction,
weight = 'combined_score'))
```

`make_drug_target` *Reformat drug-target-interaction data*

Description

Function to transform input data to required input format for `start_pipeline`. Here the data needed to define drug-target interactions is formatted. When the reformatted output is passed to `start_pipeline` as `drug_target_interaction` argument, the differential drug response score will be calculated for all the supplied drugs in `interaction_table`.

Usage

```
make_drug_target(target_molecules, interaction_table, match_on)
```

Arguments

`target_molecules`	Name of layer containing the drug targets. This name has to match the corresponding named item in the list of layers supplied to `start_pipeline`.
`interaction_table`	Data frame. Has to contain two columns. Additional columns will be ignored in the pipeline. * A column called 'drug_name' containing names or identifiers of drugs * A column with a name that matches an identifier in the layer supplied in 'target_molecules'. For example, if drugs target proteins and an identifier called 'ncbi_id' was supplied in layer building of the protein layer (`make_layer`), this column should be called 'ncbi_id' and contain the corresponding IDs of protein-drug targets. Any other ID present in the constructed layer can be used.
`match_on`	Column name of the data frame supplied in 'interaction_table' that is used for matching drugs and target nodes in the graph (e.g. 'ncbi_id').

Value

Named list of the input parameters in input format of `start_pipeline`.

Examples

```
data(drug_gene_interactions)
drug_target_interaction <- make_drug_target(target_molecules='protein',
interaction_table=drug_gene_interactions, match_on='gene_name')
```

`make_layer` *Creates individual molecular layers from raw data and unique identifiers*

Description

Helper function to transform input data to required pipeline input format. Additionally, the supplied input is checked. Allows easy conversion of raw data into the structure accepted by `start_pipeline`.

Usage

```
make_layer(
  name,
  data_group1,
  data_group2,
  identifier_group1,
  identifier_group2
)
```

Arguments

`name`	Character string. Names the layer.
`data_group1, data_group2`	Data frame containing raw molecular data of each group (each stratum). Analyzed components (e.g., genes) in columns, samples (e.g. patients) in rows.
`identifier_group1, identifier_group2`	Data frame containing component identifiers (columns) of each component (rows) in the same order as the molecular data frame of each group. These identifiers are used to (a) interconnect graphs and (b) match drugs to drug targets. Must contain a column ‘type‘ which identifies the nature of the component (e.g., "protein")

Value

Named list containing the supplied data for each group (i.e., the dataset for one layer), that can be supplied to `start_pipeline` and ‘name‘ giving the name of the layer. Each sublist contains the ‘data‘ and the ‘identifiers‘.

Examples

```
data(mrna_data)

mrna_layer <- make_layer(name="mrna", data_group1=mrna_data$group1$data,
data_group2=mrna_data$group2$data,
identifier_group1=data.frame(gene_name=mrna_data$group1$identifiers),
identifier_group2=data.frame(gene_name=mrna_data$group2$identifiers))
```

`metabolite_data` *Metabolomics data*

Description

Metabolomics analysis of breast cancer patients data sampled randomly to generate distributions similar to those reported e.g. in (Terunuma et al., 2014). The data is stratified by estrogen receptor (ER) expression status (group1 = ER+, group2 = ER-). Each group is given as a sub-list containing 'data' (raw data, metabolites in columns and samples in rows) and 'identifiers' (one column per identifier, rows in the same order as the metabolites order in 'data').

Usage

```
metabolite_data
```

Format

group1 ER+ data

- **data** raw data, metabolites in columns and samples in rows
- **identifiers** one column per identifier, rows in the same order as the metabolite order in 'data', identifiers: biochemical name, METABOLON ID, Pubchem ID

group2 ER- data

- **data** see above
- **identifiers** see above

Source

Terunuma, Atsushi et al. "MYC-driven accumulation of 2-hydroxyglutarate is associated with breast cancer prognosis." The Journal of clinical investigation vol. 124,1 (2014): 398-412. doi:10.1172/JCI71180

https://www.metabolon.com

Pubchem IDs: https://pubchem.ncbi.nlm.nih.gov

MetaboAnalyst: https://www.metaboanalyst.ca/faces/upload/ConvertView.xhtml

`metabolite_protein_interaction` *Metabolite protein interaction data*

Description

Data frame providing interactions of metabolites and proteins. The data was taken from the STITCH Database.

Usage

```
metabolite_protein_interaction
```

Format

A data frame with 3 columns.

pubchemID Pubchem IDs defining interacting metabolites

STRING_id STRING IDs defining interacting proteins

combined_score Score describing the strength of metabolite-protein interaction

Source

STITCH DB: `http://stitch.embl.de/`

Pubchem IDs: `https://pubchem.ncbi.nlm.nih.gov`

STRING DB: `https://string-db.org/`

`molnet_settings` *Create global settings variable for molnet pipeline*

Description

Function that allows creating a global ‘settings‘ variable used in the start_pipeline function. Default parameters can be changed within the function call.

Usage

```
molnet_settings(
  correlation_method = "pearson",
  print_graph_info = TRUE,
  reduction_method = "p_value",
  handling_missing_data = "all.obs",
  p_value_adjust_method = "BH",
  reduction_alpha = 0.05,
  r_squared_cutoff = 0.6,
  cut_vector = seq(0.2, 0.8, by = 0.05),
  n_threads = 1,
  parallel_chunk_size = 10^6,
  saving_path = tempdir(),
  save_individual_graphs = TRUE,
  save_combined_graphs = TRUE,
  save_drug_targets = TRUE,
  save_correlation_filename = NULL,
  python_executable = "python3",
  max_path_length = 3,
  int_score_mode = "auto",
  ...
)
```

Arguments

`correlation_method`	Correlation method used for graph generation. One of (‘pearson‘, ‘spearman‘, ‘kendall‘).
`print_graph_info`	Boolean. Print a summary of the reduced graph to console after generation?
`reduction_method`	Reduction method for reducing networks. One of ‘p_value‘, ’pickHardThreshold’ or ‘pickHardThreshold_alternative‘. Can be a single character string if the same for all layers, else a named list mapping layer names to methods. Layers may be omitted if a method is mapped to ‘default‘.
`handling_missing_data`	Specifying the handling of missing data during correlation computation. Use "all.obs" or "pairwise.complete.obs". Argument is passed to `cor`. Can be a single character string if the same for all layers, else a named list mapping layer names to methods. Layers may be omitted if a method is mapped to ‘default‘.
`p_value_adjust_method`	String of the correction method applied to p-values. Passed to p.adjust. ("holm", "hochberg", "hommel", "bonferroni",
`reduction_alpha`	A number indicating the significance value for correlation p-values during reduction. Not-significant edges are dropped.
`r_squared_cutoff`	A number indicating the desired minimum scale free topology fitting index R^2 for reduction using `pickHardThreshold`.
`cut_vector`	A vector of hard threshold cuts for which the scale free topology fit indices are to be calculated during reduction with `pickHardThreshold`.
`n_threads`	Number of threads for parallel computation of p-values during p-value reduction.
`parallel_chunk_size`	Number of p-values in smallest work unit when computing in parallel during network reduction with method ‘p_value‘.

`saving_path`	Path to save outputs of ‘molnet‘ functions. Default is a temporary directory.
`save_individual_graphs`	Boolean specifying if individual graphs should be saved during `start_pipeline`
`save_combined_graphs`	Boolean specifying if combined graphs should be saved during `start_pipeline`
`save_drug_targets`	Boolean specifying if drug targets should be saved during `start_pipeline`
`save_correlation_filename`	File name for saving correlation adjacency matrices in `generate_individual_graphs`.
`python_executable`	Path to Python executable used for computing simple paths.
`max_path_length`	Integer of maximum length of simple paths to include in computation.
`int_score_mode`	One of ‘auto‘, ‘sequential‘ or ‘ray‘. Whether to compute interaction score in parallel using the Ray python library or sequentially. When ‘auto‘ it depends on the graph sizes.
`...`	Supply additional settings.

Value

Named list of settings

Examples

```
settings <- molnet::molnet_settings(correlation_method = "spearman", max_path_length = 3,
                                    handling_missing_data = list(
                                      default = "pairwise.complete.obs",
                                      mrna = "all.obs"
                                    ),
                                    reduction_method = "p_value"
                                   )
```

`mrna_data` *mRNA expression data*

Description

mRNA analysis of breast cancer patients data from Krug et al., 2020 (data from the Clinical Proteomic Tumor Analysis Consortium (CPTAC)). The data is stratified by estrogen receptor (ER) expression status (group1 = ER+, group2 = ER-). Each group is given as a sub-list containing ’data’ (raw data, mRNAs in columns and samples in rows) and ’identifiers’ (one column per identifier, rows in the same order as the mRNA order in ’data’).

Usage

```
mrna_data
```

Format

group1 ER+ data

- **data** raw data, mRNA in columns and samples in rows
- **identifiers** one column per identifier, rows in the same order as the mRNA order in 'data', identifiers: gene name

group2 ER- data

- **data** see above
- **identifiers** see above

Source

Krug, Karsten et al. "Proteogenomic Landscape of Breast Cancer Tumorigenesis and Targeted Therapy." Cell vol. 183,5 (2020): 1436-1456.e31. doi:10.1016/j.cell.2020.10.036

`network_reduction_by_pickHardThreshold`
Reduces network based on WGCNA::pickHardThreshold function

Description

(INTERNAL) This function uses `pickHardThreshold.fromSimilarity` or an alternative implementation `pickHardThreshold_alternative` contained in this package to analyze scale free topology for multiple hard thresholds. Within the first iteration a 'coarse' cutoff is estimated. If no cutoff is found the function terminates with an error message. The second iteration determines a 'fine-grained' cutoff based on the first iterations cut estimate (+/- 0.25) in sequence steps of 0.01. All values below the cutoff will be set to NA and the reduced adjacency is returned.

Usage

```
network_reduction_by_pickHardThreshold(
  adjacency_matrix,
  RsquaredCut = 0.85,
  cutVector = seq(0.2, 0.8, by = 0.05),
  method = "pickHardThreshold"
)
```

Arguments

`adjacency_matrix`	Adjacency matrix of correlation values.
`RsquaredCut`	A number indicating the desired minimum scale free topology fitting index R^2.
`cutVector`	A vector of hard threshold cuts for which the scale free topology fit indices are to be calculated.
`method`	String. Determines whether the original implementation of `pickHardThreshold.fromSimilarity` is used ("pickHardThreshold") or the alternative implementation contained in this package `pickHardThreshold_alternative` ("pickHardThreshold_alternative").

Value

A reduced adjacency matrix of correlations with NA's inserted at positions below estimated cutoff.

Source

The original implementation of pickHardThreshold is used from `pickHardThreshold.fromSimilarity`

Examples

```
data(mrna_data)
adj_mat <- WGCNA::cor(mrna_data$group1$data)
reduced_by_PHT <- network_reduction_by_pickHardThreshold(adj_mat,
RsquaredCut = 0.1, cutVector = seq(0.2, 0.8, by = 0.05))
```

---

network_reduction_by_p_value

Reduce the the entries in an adjacency matrix by thresholding on p-values

---

Description

(INTERNAL) This function reduces an adjacency matrix of correlations. If computations are done non-parallel `corPvalueStudent` is used. If computations are done in parallel, our own parallel implementation (`corPvalueStudentParallel`) of this function is used. function to calculate Student asymptotic p-values taking the number of samples into account. P-values are adjusted using p.adjust function. The upper triangle without diagonal entries of the adjacency matrix is passed for faster computation. P-values can be adjusted using one of several methods. A significance threshold ‘alpha‘ can be set. All value entries below this threshold within the initial adjacency matrix will be set to NA. If a default cluster is registered with the ‘parallel‘ package the computation will happen in parallel automatically.

Usage

```
network_reduction_by_p_value(
  adjacency_matrix,
  number_of_samples,
  reduction_alpha = 0.05,
  p_value_adjustment_method = "BH",
  parallel_chunk_size = 10^6
)
```

Arguments

`adjacency_matrix`
: An adjacency matrix of correlation values.

`number_of_samples`
: The number of samples used to calculate the correlation matrix.

`reduction_alpha`
: A number indicating the alpha value applied for thresholding

`p_value_adjustment_method`
: A string of the correction method applied to p-values. Passed to stats::p.adjust().

`parallel_chunk_size`
: Number of p-values in smallest work unit when computing in parallel.

Value

A reduced adjacency matrix with NA's at martix entries with p-values below threshold.

Source

corPvalueStudent

Examples

```
adj_mat <- matrix(rnorm(36),nrow=6)
sum(is.na(adj_mat)) # before reduction
reduced_by_p_value_matrix <- network_reduction_by_p_value(adjacency_matrix=adj_mat,
                             number_of_samples=200, reduction_alpha = 0.05,
                             p_value_adjustment_method = "BH")
sum(is.na(reduced_by_p_value_matrix)) # after reduction
```

`phosphoprotein_data` *Phosphosite data*

Description

Phosphosite analysis of breast cancer patients data from Krug et al., 2020 (data from the Clinical Proteomic Tumor Analysis Consortium (CPTAC)). The data is stratified by estrogen receptor (ER) expression status (group1 = ER+, group2 = ER-). Each group is given as a sub-list containing 'data' (raw data, phosphosites in columns and samples in rows) and 'identifiers' (one column per identifier, rows in the same order as the phosphosite order in 'data').

Usage

```
phosphoprotein_data
```

Format

group1 ER+ data

- **data** raw data, phosphosites in columns and samples in rows
- **identifiers** one column per identifier, rows in the same order as the phoshosite order in 'data', identifiers: Phosphosite ID, RefSeq ID, gene name

group2 ER- data

- **data** see above
- **identifiers** see above

Source

Krug, Karsten et al. "Proteogenomic Landscape of Breast Cancer Tumorigenesis and Targeted Therapy." Cell vol. 183,5 (2020): 1436-1456.e31. doi:10.1016/j.cell.2020.10.036

`pickHardThreshold_alternative`

Alternative implementation of WGCNA::pickHardThreshold

Description

(INTERNAL) Alternative implementation of `pickHardThreshold` to fit to the needs of this package. Most importantly the function was simplified to only apply to the use case of finding a cut-off value to reduce a correlation matrix. The following changes were applied in comparison to the original function: * The function __always__ assumes similarity matrices (i.e. correlation matrices) as input * Additional settings have been removed (‘dataIsExpr‘, ‘moreNetworkConcepts‘, ‘removeFirst‘, ‘corFnc‘, ‘corOptions‘, ‘nBreaks‘) * The function uses `scaleFreeFitIndex_alternative` for fit index calculation * Print prompts and additional metrics were removed * An error message that reports the lowest R-squared computed in case this value did not satisfy the `RsquaredCut` value was added.

Description by `pickHardThreshold`:Analysis of scale free topology for multiple hard thresholds. The aim is to help the user pick an appropriate threshold for network construction.

Usage

```
pickHardThreshold_alternative(
  data,
  RsquaredCut = 0.85,
  cutVector = seq(0.1, 0.9, by = 0.05)
)
```

Arguments

`data`	Similarity (correlation) matrix. With entries between 0 and 1 (i.e. absolute values of correlation matrix)
`RsquaredCut`	desired minimum scale free topology fitting index
`cutVector`	a vector of hard threshold cuts for which the scale free topology fit indices are to be calculated.

Value

estimate of an appropriate hard-thresholding cut: the lowest cut for which the scale free topology fit exceeds `RsquaredCut`. If is below `RsquaredCut` for all cuts, an error is thrown.

Source

`pickHardThreshold` and `scaleFreeFitIndex`

Examples

```
adjacency_matrix <- matrix(rnorm(36),nrow=6)
diag(adjacency_matrix) <- 1
RsquaredCut <- 0.001
cutVector <- seq(0.2, 0.8, by = 0.05)

cutEstimate_coarse <- pickHardThreshold_alternative(abs(adjacency_matrix), RsquaredCut,
cutVector)
```

`protein_data` *Protein data*

Description

Protein analysis of breast cancer patients data from Krug et al., 2020 (data from the Clinical Proteomic Tumor Analysis Consortium (CPTAC)). The data is stratified by estrogen receptor (ER) expression status (group1 = ER+, group2 = ER-). Each group is given as a sub-list containing 'data' (raw data, proteins in columns and samples in rows) and 'identifiers' (one column per identifier, rows in the same order as the protein order in 'data').

Usage

```
protein_data
```

Format

group1 ER+ data

- **data** raw data, protein in columns and samples in rows
- **identifiers** one column per identifier, rows in the same order as the protein order in 'data', identifiers: RefSeq ID, gene name

group2 ER- data

- **data** see above
- **identifiers** see above

Source

Krug, Karsten et al. "Proteogenomic Landscape of Breast Cancer Tumorigenesis and Targeted Therapy." Cell vol. 183,5 (2020): 1436-1456.e31. doi:10.1016/j.cell.2020.10.036

`return_errors` *Return detected errors*

Description

Throws an error in case errors have been passed to the function. Messages describing the detected errors are printed.

Usage

```
return_errors(errors)
```

Arguments

`errors`	Character string vector containing error messages.

Examples

```
layer <- molnet::layers_example[[2]]
return_errors(check_layer(layer))
```

`sample_size` *Sample size for correlation computation*

Description

(INTERNAL) Depending on how missing data is handled in correlation matrix computation, the number of samples used is returned. If ‘all.obs‘ is used the number of rows (i.e. samples) of the orignal data is returned. If ‘pairwise.complete.obs‘ is used the crossproduct of a matrix indicating the non-NA values is returned as matrix. This implementation was adopted from `corAndPvalue`.

Usage

```
sample_size(x, use)
```

Arguments

`x`	matrix of measurement data used for correlation computation
`use`	string indicating handling of missing data. Can be 'all.obs' for all observations or 'pairwise.complete.obs' for pairwise observations

Value

For 'all.obs' returns an integer indicating the number of samples in the supplied matrix (i.e. number of rows). For 'pairwise.complete.obs' returns a matrix in the same size of the correlation matrix indicating the number of samples for each correlation calculation.

Source

Method to calculate samples in ‘pairwise.complete.obs‘ adopted and improved from `corAndPvalue`

`scaleFreeFitIndex_alternative` *Alternative implementation of WGCNA::scaleFreeFitIndex*

Description

(INTERNAL) This function is a copy of `scaleFreeFitIndex` with minor changes introduced to fit the needs of this package.

Description by `scaleFreeFitIndex`: The function scaleFreeFitIndex calculates several indices (fitting statistics) for evaluating scale free topology fit. The input is a vector (of connectivities) k. Next k is discretized into nBreaks number of equal- width bins. Let's denote the resulting vector dk. The relative frequency for each bin is denoted p.dk.

The entire original function code is contained in the function and deleted lines are commented out using `#--` to show changes.

Usage

```
scaleFreeFitIndex_alternative(k, nBreaks = 10, removeFirst = FALSE)
```

Arguments

`k`	numeric vector whose components contain non-negative values
`nBreaks`	positive integer. This determines the number of equal width bins.
`removeFirst`	logical. If TRUE then the first bin will be removed.

Value

the model fitting index (R.squared) from the following model lm(log.p.dk ~ log.dk)

Source

`scaleFreeFitIndex`

`set_cluster` *Create and register cluster*

Description

(INTERNAL) Helper function to create and register a cluster for parallel computation of p-value reduction

Usage

```
set_cluster(n_threads)
```

Arguments

`n_threads`	number of nodes in the cluster

`shutdown_cluster` *Shutdown cluster and remove corresponding connections*

Description

(INTERNAL) Run this if the pipeline fails during parallel computation to clean the state. If a cluster is registered, this functions stops it and removes corresponding connections. Ignores errors. Has no effect if no cluster is registered.

Usage

```
shutdown_cluster()
```

`start_pipeline` *Execute all molnet-pipeline steps sequentially*

Description

This wrapper function executes all necessary steps to generate differential drug response scores from the formatted input data. The following input data is required (and detailed below): * Layers of stratified molecular data. * Additional connections between the layers. * Interactions between drugs and nodes in the network. * Settings for pipeline execution.

Usage

```
start_pipeline(
  layers,
  inter_layer_connections,
  drug_target_interaction,
  settings
)
```

Arguments

`layers`	Named list with different network layers containing data and identifiers for both groups. The required input format is a list with names corresponding to the content of the respective layer (e.g., "protein"). Each named element has to contain the molecular data and corresponding identifiers formatted by `make_layer`.
`inter_layer_connections`	A list with specified inter-layer connections. This list contains one or more elements defining individual inter-layer connections created by `make_connection`.
`drug_target_interaction`	A list specifying drug-target interactions for drug response score computation. The required input format of this list is created by `make_drug_target`. The drug response score is calculated for all drugs contained in this object.
`settings`	A named list containing pipeline settings. The settings list has to be initialized by `molnet_settings`. Items in the named list can be adjusted as desired.

Details

As this function runs through all steps of the molnet-pipeline it can take a long to complete, especially if the supplied molecular data is in large dimensions. Several prompts will be printed to supply information on how the pipeline is proceeding. Calculation of the interaction score by `interaction_score` requires saving large-scale graphs to file and calls a python script. This handover may take time.

Eventually a data frame is returned containing the supplied drug name and its associated differential drug response score computed by molnet.

Value

Data frame containing drug name and associated differential drug response score. If no target is found for a specific drug, `NA` is returned as a score. If Python is not installed or the interaction score computation fails for some other reason, NULL is returned instead.

Examples

```
data(drug_gene_interactions)
data(layers_example)
inter_layer_connections = list(make_connection(from = 'mrna',
to = 'protein',
connect_on = 'gene_name'))
```

```
drug_target_interaction <- make_drug_target(target_molecules='protein',
interaction_table=drug_gene_interactions,
match_on='gene_name')
settings <- molnet_settings(handling_missing_data = list(default =
"pairwise.complete.obs",mrna = "all.obs"),
save_individual_graphs = FALSE,
save_combined_graphs = FALSE,
save_drug_targets = FALSE,
python_executable = "python3")


start_pipeline(layers_example, inter_layer_connections, drug_target_interaction, settings)
```

`target_edge_list` *Edges adjacent to target nodes*

Description

(INTERNAL) Based on the supplied graph and target nodes this function returns a list of edges that are directly adjacent to target nodes. These edges can be used for further computation to find differential scores in the networks.

Usage

```
target_edge_list(graph, target_nodes, group)
```

Arguments

`graph`	Combined graph (iGraph graph object) for a specific group
`target_nodes`	Data frame. Has column 'node_id' (unique node IDs in the iGraph graph object that are targeted by drugs) and columns 'group1' and 'group2' (boolean values specifying whether the node is contained in the combined graph of the group)
`group`	Character. Indicates which group is analyzed.

Value

An edge list as a data frame.

write_interaction_score_input

Write edge lists and combined graphs to files

Description

(INTERNAL) Writes the combined graphs and the drug target edge lists to files for passing them to the python interaction score script. Graphs are saved as ‘gml‘ file. Edgelists are saved as ‘tsv‘ file.

Usage

```
write_interaction_score_input(
  combined_graphs,
  drug_target_edgelists,
  saving_path
)
```

Arguments

`combined_graphs`	A named list (elements ‘group1‘ and ‘group2‘). Each element contains the entire combined network (layers + inter-layer connections) as iGraph graph object.
`drug_target_edgelists`	A named list (elements ‘group1‘ and ‘group2‘). Each element contains the list of edges to be considered in the interaction score calculation as data frame (columns ‘from‘, ‘to‘ and ‘weight‘)
`saving_path`	Directory to write to

Value

Does not return value, but writes to .tsv.

Index