Package 'DrDimont'

January 20, 2025

Type Package

Title Drug Response Prediction from Differential Multi-Omics Networks

Version 0.1.4

Description While it has been well established that drugs affect and help patients differently, personalized drug response predictions remain challenging. Solutions based on single omics measurements have been proposed, and networks provide means to incorporate molecular interactions into reasoning. However, how to integrate the wealth of information contained in multiple omics layers still poses a complex problem.
We present a novel network analysis pipeline, DrDimont, Drug response prediction from Differential analysis of multi-omics networks. It allows for comparative conclusions between two conditions and translates them into differential drug response predictions. DrDimont focuses on molecular interactions. It establishes condition-specific networks from correlation within an omics layer that are then reduced and combined into heterogeneous, multi-omics molecular networks. A novel semi-local, path-based integration step ensures integrative conclusions. Differential predictions are derived from comparing the condition-specific integrated networks. DrDimont's predictions are explainable, i.e., molecular differences that are the source of high differential drug scores can be retrieved. 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 7.2.1

VignetteBuilder knitr

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

Suggests rmarkdown, knitr

Depends R (>= 3.5.0)

NeedsCompilation no

Author Katharina Baum [cre] (<https://orcid.org/0000-0001-7256-0566>),
Pauline Hiort [aut] (<https://orcid.org/0000-0002-3530-7358>),
Julian Hugo [aut] (<https://orcid.org/0000-0003-3355-1071>),
Spoorthi Kashyap [aut] (<https://orcid.org/0000-0002-5474-8183>),
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 2022-09-23 15:40:02 UTC

Contents

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 DrDimont pipeline.

Usage

```
check_input(layers, inter_layer_connections, drug_target_interactions)
```

Arguments

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

Value

Character string vector containing error messages.

Examples

```
data(layers_example)
data(metabolite_protein_interactions)
data(drug_gene_interactions)
data

all_layers <- layers_example

all_inter_layer_connections = list(
    make_connection(from='mrna', to='protein', connect_on='gene_name', weight=1),
    make_connection(from='protein', to='phosphosite', connect_on='gene_name', weight=1),
    make_connection(from='protein', to='metabolite',
    connect_on=metabolite_protein_interactions, weight='combined_score'))

all_drug_target_interactions <- make_drug_target(
                                    target_molecules="protein",
                                    interaction_table=drug_gene_interactions,
                                    match_on="gene_name")

return_errors(check_input(layers=all_layers,
    inter_layer_connections=all_inter_layer_connections,
    drug_target_interactions=all_drug_target_interactions))
```

combined_graphs_example *Combined graphs*

Description

Exemplary intermediate pipeline output: Combined graphs example data built by `generate_combined_graphs`. Combined graphs were built using the individual_graphs_example and:

Usage

```
combined_graphs_example
```

Format

A named list with 2 items.

graphs A named list with two groups.

groupA Graph associated with 'groupA'

groupB Graph associated with 'groupB'

annotations A data frame of mappings of assigned node IDs to the user-provided component identifiers for all nodes in 'groupA' and 'groupB' together and all layers

both Data frame

Details

```
inter_layer_connections = list( make_connection(from='mrna', to='protein', connect_on='gene_name',
weight=1), make_connection(from='protein', to='phosphosite', connect_on='gene_name',
weight=1), make_connection(from='protein', to='metabolite', connect_on=metabolite_protein_interactior
weight='combined_score'))
```

A subset of the original data by Krug et al. (2020) and randomly sampled metabolite data from `layers_example` was used to generate the correlation matrices, individual graphs and combined graphs. They were created from data stratified by estrogen receptor (ER) status: 'groupA' contains data of ER+ patients and 'groupB' of ER- patients.

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

compute_correlation_matrices *Computes correlation matrices for specified network layers*

Description

Constructs and returns a correlation/adjacency matrices for each network layer and each group. The adjacency matrix of correlations is computed using cor. The handling of missing data can be specified. Optionally, the adjacency matrices of the correlations can be saved. Each node is mapped to the biological identifiers given in the layers and the mapping table is returned as 'annotations'.

Usage

```
compute_correlation_matrices(layers, settings)
```

Arguments

layers	[list] Named list with different network layers containing data and identifiers for both groups (generated from make_layer)
settings	[list] A named list containing pipeline settings. The settings list has to be initialized by drdimont_settings. Items in the named list can be adjusted as desired.

Value

A nested named list with first-level elements 'correlation_matrices' and 'annotations'. The second level elements are 'groupA' and 'groupB' (and 'both' at 'annotations'). These contain a named list of matrix objects ('correlation_matrices') and data frames ('annotations') mapping the graph node IDs to biological identifiers. The third level elements are the layer names given by the user.

Examples

```
example_settings <- drdimont_settings(
                        handling_missing_data=list(
                            default="all.obs"))

# mini example with reduced mRNA layer for shorter runtime:
data(mrna_data)
reduced_mrna_layer <- make_layer(name="mrna",
                           data_groupA=mrna_data$groupA[1:5,2:6],
                           data_groupB=mrna_data$groupB[1:5,2:6],
                  identifiers_groupA=data.frame(gene_name=mrna_data$groupA$gene_name[1:5]),
                  identifiers_groupB=data.frame(gene_name=mrna_data$groupB$gene_name[1:5]))

example_correlation_matrices <- compute_correlation_matrices(
                                      layers=list(reduced_mrna_layer),
                                      settings=example_settings)
```

```
# to run all layers use layers=layers_example from data(layers_example)
# in compute_correlation_matrices()
```

---

compute_drug_response_scores *Calculate drug response score*

Description

This function takes the differential graph (generated in generate_differential_score_graph), 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 mean or median of all differential scores of the edges adjacent to all drug target nodes of a particular drug.

Usage

```
compute_drug_response_scores(differential_graph, drug_targets, settings)
```

Arguments

differential_graph	iGraph graph object containing differential scores for all edges. (output of generate_differential_score_graph)
drug_targets	[list] Named list containing two elements ('target_nodes' and 'drugs_to_target_nodes'). 'targets' from 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.
settings	[list] A named list containing pipeline settings. The settings list has to be initialized by drdimont_settings. Items in the named list can be adjusted as desired.

Value

Data frame containing drug name and associated differential (integrated) drug response score

Examples

```
data(drug_target_edges_example)
data(differential_graph_example)

example_settings <- drdimont_settings()

example_drug_response_scores <- compute_drug_response_scores(
                                    differential_graph=differential_graph_example,
```

```
drug_targets=drug_target_edges_example$targets,
settings=example_settings)
```

`correlation_matrices_example`
Correlation matrices

Description

Exemplary intermediate pipeline output: Correlation matrices example data built by `compute_correlation_matrices` using layers_example data and settings:

Usage

```
correlation_matrices_example
```

Format

A named list with 2 items.

correlation_matrices A named list with two groups.

- **groupA** Correlation matrices associated with ‘groupA‘
  - **mrna** Correlation matrix
  - **protein** Correlation matrix
  - **phosphosite** Correlation matrix
  - **metabolite** Correlation matrix
- **groupB** same structure as ‘groupA‘

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

- **groupA** Annotations associated with ‘groupA‘
  - **mrna** Data frame
  - **protein** Data frame
  - **phosphosite** Data frame
  - **metabolite** Data frame
- **groupB** same structure as ‘groupA‘
- **both** same structure as ‘groupA‘

Details

```
settings <- drdimont_settings( handling_missing_data=list( default="pairwise.complete.obs",
mrna="all.obs"))
```

A subset of the original data from Krug et al. (2020) and randomly sampled metabolite data in layers_example was used to generate the correlation matrices. They were created from data stratified by estrogen receptor (ER) status: ‘groupA‘ contains data of ER+ patients and ‘groupB‘ of ER- patients.

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

`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_interactions'. Returns list of node ids and list of adjacent edges.

Usage

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

Arguments

`graphs`	[list] A named list with elements 'groupA' and 'groupB' containing the combined graphs of each group as iGraph object ('graphs' from output of `generate_combined_graphs`)
`annotations`	[list] List of data frames that map node IDs to identifiers. Contains 'both' with unique identifiers across the whole data (output of `generate_combined_graphs`)
`drug_target_interactions`	[list] Named list specifying drug target interactions for drug response score computation
`settings`	[list] A named list containing pipeline settings. The settings list has to be initialized by `drdimont_settings`. Items in the named list can be adjusted as desired.

Value

A named list with elements 'targets' and 'edgelists'. '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 'groupA' and 'groupB' (bool 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. 'edgelists' contains elements 'groupA' and 'groupB' containing each a list of edges adjacent to drug target nodes.

Examples

```
data(drug_gene_interactions)
data(combined_graphs_example)

example_settings <- drdimont_settings()
```

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

example_drug_target_edges <- determine_drug_targets(
                                   graphs=combined_graphs_example$graphs,
                                   annotations=combined_graphs_example$annotations,
                                drug_target_interactions=example_drug_target_interactions,
                                   settings=example_settings)
```

`differential_graph_example`
Differential graph

Description

Exemplary intermediate pipeline output: Differential score graph example data built by generate_differential_score_gr using the interaction_score_graphs_example. Consists of one graph containing edge attributes: the differential correlation values as 'differential_score' and the differential interaction score as 'differential_interaction_score'.

Usage

```
differential_graph_example
```

Format

An iGraph graph object.

Details

A subset of the original data by Krug et al. (2020) and randomly sampled metabolite data from layers_example was used to generate the correlation matrices, individual graphs and combined graphs. They were created from data stratified by estrogen receptor (ER) status: 'groupA' contains data of ER+ patients and 'groupB' of ER- patients.

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

drdimont_settings *Create global settings variable for DrDimont pipeline*

Description

Allows creating a global 'settings' variable used in DrDimont's run_pipeline function and stepwise execution. Default parameters can be changed within the function call.

Usage

```
drdimont_settings(
  saving_path = tempdir(),
  save_data = FALSE,
  correlation_method = "spearman",
  handling_missing_data = "all.obs",
  reduction_method = "pickHardThreshold",
  r_squared_cutoff = 0.85,
  cut_vector = seq(0.2, 0.8, by = 0.01),
  mean_number_edges = NULL,
  edge_density = NULL,
  p_value_adjustment_method = "BH",
  reduction_alpha = 0.05,
  n_threads = 1,
  parallel_chunk_size = 10^6,
  print_graph_info = TRUE,
  conda = FALSE,
  max_path_length = 3,
  int_score_mode = "auto",
  cluster_address = "auto",
  median_drug_response = FALSE,
  absolute_difference = FALSE,
  ...
)
```

Arguments

saving_path	[string] Path to save intermediate output of DrDimont's functions. Default is temporary folder.
save_data	[bool] Save intermediate data such as correlation_matrices, individual_graphs, etc. during exectution of DrDimont. (default: FALSE)
correlation_method	["pearson"|"spearman"|"kendall"] Correlation method used for graph generation. Argument is passed to cor. (default: spearman)
handling_missing_data	["all.obs"|"pairwise.complete.obs"] Method for handling of missing data during correlation matrix computation. 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, e.g, handling_missing_data=list(mrna="all.obs", protein="pairwise.complete.ob Layers may be omitted if a method is mapped to 'default', e.g, handling_missing_data=list(default= (default: all.obs)

reduction_method

["pickHardThreshold"|"p_value"] Reduction method for reducing networks. '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 pickHardThreshold). Can be a single character string if the same for all layers, else a named list mapping layer names to methods (see handling_missing_data setting). Layers may be omitted if a method is mapped to 'default'. (default: pickHardThreshold)

r_squared_cutoff

pickHardThreshold setting: [float|named list] Minimum scale free topology fitting index R^2 for reduction using pickHardThreshold. Can be a single float number if the same for all layers, else a named list mapping layer names to a cutoff (see handling_missing_data setting) or a named list in a named list mapping groupA or groupB and layer names to a cutoff, e.g., r_squared_cutoff=list(groupA=list(mrna= protein=0.8), groupB=list(mrna=0.9, protein=0.85)). Layers/groups may be omitted if a cutoff is mapped to 'default'. (default: 0.85)

cut_vector

pickHardThreshold setting: [sequence of float|named list] Vector of hard threshold cuts for which the scale free topology fit indices are calculated during reduction with pickHardThreshold. Can be a single regular sequence if the same for all layers, else a named list mapping layer names to a cut vector or a named list in a named list mapping groupA or groupB and layer names to a cut vector (see r_squared_cutoff setting). Layers/groups may be omitted if a vector is mapped to 'default'. (default: seq(0.2, 0.8, by = 0.01))

mean_number_edges

pickHardThreshold setting: [int|named list] Maximal mean number edges threshold to find a suitable edge weight cutoff employing pickHardThreshold to reduce the network to at most the specified mean number of edges. Can be a single int number if the same for all layers, else a named list mapping layer names to a mean number of edges or a named list in a named list mapping groupA or groupB and layer names to a cutoff (see r_squared_cutoff setting). Attention: This parameter overwrites the 'r_squared_cutoff' and 'edge_density' parameters if not set to NULL. (default: NULL)

edge_density

pickHardThreshold setting: [float|named list] Maximal network edge density to find a suitable edge weight cutoff employing pickHardThreshold to reduce the network to at most the specified edge density. Can be a single float number if the same for all layers, else a named list mapping layer names to a mean number of edges or a named list in a named list mapping groupA or groupB and layer names to a cutoff (see r_squared_cutoff setting). Attention: This parameter overwrites the 'r_squared_cutoff' parameter if not set to NULL. (default: NULL)

p_value_adjustment_method

p_value setting: ["holm"|"hochberg"|"hommel"|"bonferroni"|"BH"|"BY"|"fdr"|"none"] Correction method applied to p-values. Passed to p.adjust. (default: "BH")

`reduction_alpha`
: p_value setting: [float] Significance value for correlation p-values during reduction. Not-significant edges are dropped. (default: 0.05)

`n_threads`
: p_value setting: [int] Number of threads for parallel computation of p-values during p-value reduction. (default: 1)

`parallel_chunk_size`
: p_value setting: [int] Number of p-values in smallest work unit when computing in parallel during network reduction with method ‘p_value‘. (default: 10^6)

`print_graph_info`
: [bool] Print summary of the reduced graph to the console after network generation. (default: TRUE)

`conda`
: [bool] Python installation in conda environment. Set TRUE if Python is installed with conda. (default: FALSE)

`max_path_length`
: [int] Integer of maximum length of simple paths to include in the `generate_interaction_score_graphs` computation. (default: 3)

`int_score_mode`
: ["auto"|"sequential"|"ray"] Interaction score sequential or parallel ("ray") computation. For parallel computation the Python library Ray ist used. When set to ‘auto‘ computation depends on the graph sizes. (default: "auto")

`cluster_address`
: [string] Local node IP-address of Ray if executed on a cluster. On a cluster: Start ray with `ray start --head --num-cpus 32` on the console before DrDimont execution. It should work with "auto", if it does not specify IP-address given by the `ray start` command. (default: "auto")

`median_drug_response`
: [bool] Computation of median (instead of mean) of a drug’s targets differential scores (default: FALSE)

`absolute_difference`
: [bool] Computation of drug response scores based on absolute differential scores (instead of the actual differential scores) (default: FALSE)

`...`
: Supply additional settings.

Value

Named list of the settings for the pipeline

Examples

```
settings <- drdimont_settings(
                correlation_method="spearman",
                handling_missing_data=list(
                    default="pairwise.complete.obs",
                    mrna="all.obs"),
                reduction_method="pickHardThreshold",
                max_path_length=3)
```

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.

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_scores_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 compute_drug_response_scores using differential_graph_example, drug_target_edges_example and

Usage

```
drug_response_scores_example
```

Format

Data frame with two columns

drug_name Names of drugs

drug_response_scores Associated differential drug response scores

Details

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

A subset of the original data by Krug et al. (2020) and randomly sampled metabolite data from `layers_example` was used to generate the correlation matrices, individual graphs and combined graphs, interaction score graphs and differential score graph. They were created from data stratified by estrogen receptor (ER) status: ‘groupA‘ contains data of ER+ patients and ‘groupB‘ of ER- patients. Drug-gene interactions 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_target_edges_example

Drug target nodes in combined network

Description

Exemplary intermediate pipeline output: Drug targets detected in the combined graphs. A named list with elements ‘targets‘ and ‘edgelists‘. This was created with `determine_drug_targets` using the combined_graphs_example and:

Usage

```
drug_target_edges_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 ‘groupA‘ and ‘groupB‘ (bool 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.

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

Details

`drug_target_interactions <- make_drug_target(target_molecules='protein', interaction_table=drug_gene_ match_on='gene_name')`

Drug-gene interactions to calculate this output were used from The Drug Gene Interaction Database.

Source

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

`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`	[list] A named list (elements 'groupA' and 'groupB'). Each element contains a list of iGraph objects ('graphs' from output of `generate_individual_graphs`).
`annotations`	[list] A named list (elements 'groupA', 'groupB' and 'both'). Each element contains a list of data frames mapping each node IDs to identifiers. 'both' contains unique identifiers across the whole data. ('annotations' from output of `generate_individual_graphs`)
`inter_layer_connections`	[list] Named list with specified inter-layer connections. Names are layer names and elements are connections (make_connection).
`settings`	[list] A named list containing pipeline settings. The settings list has to be initialized by `drdimont_settings`. Items in the named list can be adjusted as desired.

Value

A named list (elements 'graphs' and sub-elements '$groupA' and '$groupB', and 'annotations' and sub-element 'both'). Contains the igraph objects of the combined network and their annotations for both groups.

Examples

```
data(individual_graphs_example)
data(metabolite_protein_interactions)

example_inter_layer_connections = list(make_connection(from='mrna', to='protein',
                                        connect_on='gene_name', weight=1),
                                    make_connection(from='protein', to='phosphosite',
                                        connect_on='gene_name', weight=1),
                                    make_connection(from='protein', to='metabolite',
                                        connect_on=metabolite_protein_interactions,
                                        weight='combined_score'))

example_settings <- drdimont_settings()

example_combined_graphs <- generate_combined_graphs(
                                graphs=individual_graphs_example$graphs,
                                annotations=individual_graphs_example$annotations,
                                inter_layer_connections=example_inter_layer_connections,
                                settings=example_settings)
```

---

generate_differential_score_graph

Compute difference of interaction score of two groups

Description

Computes the absolute difference of interaction scores between the two groups. Returns a single graph with the differential score and the differential interaction score as edge attributes. The interaction score is computed by generate_interaction_score_graphs.

Usage

```
generate_differential_score_graph(interaction_score_graphs, settings)
```

Arguments

interaction_score_graphs	[list] Named list with elements ‘groupA‘ and ‘groupB‘ containing iGraph objects with weight and interaction_weight as edge attributes (output of generate_interaction_score_gr
settings	[list] A named list containing pipeline settings. The settings list has to be initialized by drdimont_settings. Items in the named list can be adjusted as desired.

Value

iGraph object with ‘differential_score‘ and ‘differential_interaction_score‘ as edge attributes

Examples

```
data(interaction_score_graphs_example)

example_settings <- drdimont_settings()

example_differential_score_graph <- generate_differential_score_graph(
                              interaction_score_graphs=interaction_score_graphs_example,
                                        settings=example_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 values of the 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.

Usage

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

Arguments

`correlation_matrices`	[list] List of correlation matrices generated with codecompute_correlation_matrices
`layers`	[list] Named list with different network layers containing data and identifiers for both groups (generated from `make_layer`)
`settings`	[list] A named list containing pipeline settings. The settings list has to be initialized by `drdimont_settings`. Items in the named list can be adjusted as desired.

Value

A nested named list with first-level elements ‘graphs‘ and ‘annotations‘. The second level elements are ‘groupA‘ and ‘groupB‘ (and ‘both‘ at ‘annotations‘). These contain a list of iGraph objects (‘graphs‘) and data frames (‘annotations‘) mapping the graph node IDs to biological identifiers. The third level elements are layer names given by the user.

Examples

```
data(layers_example)
data(correlation_matrices_example)

example_settings <- drdimont_settings(
                        handling_missing_data=list(
                            default="pairwise.complete.obs",
                            mrna="all.obs"),
                        reduction_method="pickHardThreshold",
                        r_squared=list(default=0.65, metabolite=0.1),
                        cut_vector=list(default=seq(0.2, 0.5, 0.01)))

example_individual_graphs <- generate_individual_graphs(
                                correlation_matrices=correlation_matrices_example,
                                layers=layers_example,
                                settings=example_settings)

graph_metrics(example_individual_graphs$graphs$groupA$mrna)
graph_metrics(example_individual_graphs$graphs$groupB$mrna)
```

`generate_interaction_score_graphs`

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 in 'tsv' format) to files and calls a Python script for calculating the interaction scores. Output files written by the Python script are two graphs in 'gml' format containing the interaction score as an additional 'interaction_weight' edge attribute. These are loaded and returned in a named list. ATTENTION: Data exchange via files is mandatory and takes a long time 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 the parameter 'max_path_length' in `drdimont_settings` to a large value and only consider this step if your graphs have approximately 2 million edges or less. Computation is initiated by `calculate_interaction_score`. The Python script is parallelized using Ray. Use the `drdimont_settings` parameter '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. Use DrDimont's `install_python_dependencies` to install a virtual Python or conda environment containing the required Python packages. You can use the parameter 'conda' in `drdimont_settings` to specify if Python packages were installed with conda ('conda=TRUE'), else a virtual environment installed with pip is assumed (default: 'conda=FALSE').

Usage

```
generate_interaction_score_graphs(graphs, drug_target_edgelists, settings)
```

Arguments

graphs	[list] A named list with elements ‘groupA‘ and ‘groupB‘ containing the combined graphs of each group as iGraph object (‘graphs‘ from output of generate_combined_graphs)
drug_target_edgelists	[list] A named list (elements ‘groupA‘ and ‘groupB‘). Each element contains the list of edges adjacent to drug targets as a data frame (columns ‘from‘, ‘to‘ and ‘weight‘). ‘edgelists‘ from output of determine_drug_targets
settings	[list] A named list containing pipeline settings. The settings list has to be initialized by drdimont_settings. Items in the named list can be adjusted as desired.

Value

A named list (elements ‘groupA‘ and ‘groupB‘). Each element contains an iGraph object containing the interaction scores as interaction_weight attributes.

Examples

```
data(combined_graphs_example)
data(drug_target_edges_example)

example_settings <- drdimont_settings()


example_interaction_score_graphs <- generate_interaction_score_graphs(
                                        graphs=combined_graphs_example$graphs,
                                drug_target_edgelists=drug_target_edges_example$edgelists,
                                        settings=example_settings)
```

---

individual_graphs_example

Individual graphs

Description

Exemplary intermediate pipeline output: Individual graphs example data built by generate_individual_graphs. Graphs were created from correlation_matrices_example and reduced by the ‘pickHardThreshold‘ reduction method. Used settings were:

Usage

```
individual_graphs_example
```

Format

A named list with 2 items.

graphs A named list with two groups.

- **groupA** Graphs associated with ‘groupA‘
  - **mrna** Graph
  - **protein** Graph
  - **phosphosite** Graph
  - **metabolite** Graph
- **groupB** same structure as ‘groupA‘

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

- **groupA** Annotations associated with ‘groupA‘
  - **mrna** Data frame
  - **protein** Data frame
  - **phosphosite** Data frame
  - **metabolite** Data frame
- **groupB** same structure as ‘groupA‘
- **both** same structure as ‘groupA‘

Details

```
settings <- drdimont_settings( reduction_method=list(default="pickHardThreshold"),
r_squared=list( default=0.8, groupA=list(metabolite=0.45), groupB=list(metabolite=0.15)),
cut_vector=list( default=seq(0.3, 0.7, 0.01), metabolite=seq(0.1, 0.65, 0.01)))
```

A subset of the original data by Krug et al. (2020) and randomly sampled metabolite data from `layers_example` was used to generate the correlation matrices and individual graphs. They were created from data stratified by estrogen receptor (ER) status: ‘groupA‘ contains data of ER+ patients and ‘groupB‘ of ER- patients.

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 pip (default) or conda as specified to install all required Python modules. The Python packages are installed into a virtual Python or conda environment called 'r-DrDimont'. The following requirements are installed: numpy, tqdm, python-igraph and ray. The environment is created with reticulate.

Usage

```
install_python_dependencies(package_manager = "pip")
```

Arguments

package_manager	["pip"|"conda"] Package manager to use (default: pip)

Value

No return value, called to install python dependencies

---

interaction_score_graphs_example

Interaction score graphs

Description

Exemplary intermediate pipeline output: Interaction score graphs example data built by generate_interaction_score_gra using combined_graphs_example and drug_target_edges_example. A named list (elements 'groupA' and 'groupB'). Each element contains an iGraph object containing edge attributes: the correlation values as 'weight' and the interaction score as 'interactionweight'.

Usage

```
interaction_score_graphs_example
```

Format

A named list with 2 items.

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

groupB

Details

A subset of the original data by Krug et al. (2020) and randomly sampled metabolite data from layers_example was used to generate the correlation matrices, individual graphs and combined graphs. They were created from data stratified by estrogen receptor (ER) status: 'groupA' contains data of ER+ patients and 'groupB' of ER- patients. Drug-gene interactions 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/

`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:

groupA Data associated with 'groupA'

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

groupB Data associated with 'groupB'

- **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, phosphosite and metabolite layers. The mRNA, protein and phosphosite data was adapted and reduced 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). The 'data' elements contain the raw data with samples as columns and molecular entities as rows. The 'identifiers' elements contain layer specific identifiers for the molecular entities, e.g, gene_name.

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

make_connection *Specify connection between two individual layers*

Description

Helper function to transform input data to the required pipeline input format. This helper function creates a list that specifies the connection between two layers. The connection can be based on IDs present in the identifiers of both layer or an interaction table containing a mapping of the connections and edge weights. Additionally, the supplied input is checked. Allows easy conversion of raw data into the structure accepted by `run_pipeline`.

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

Usage

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

Arguments

`from`	[string] Name of the layer from which the connection should be established
`to`	[string] Name of the layer to which the connection should be established
`connect_on`	[string|table] 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. The table has to contain an additional column specifying the weight between two components/nodes (see 'weight' argument)
`weight`	[int|string] Specifies the edge weight between the layers. This can be supplied as a number applied to every connection or a column name of the interaction table. Fixed weight: A umber 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. (default: 1)
`group`	["A"|"B"|"both"] Group for which to apply the connection. One of 'both', 'A' or 'B'. (default: "both")

Value

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

Examples

```
data(metabolite_protein_interactions)

example_inter_layer_connections = list(make_connection(from='mrna', to='protein',
                                        connect_on='gene_name', weight=1),
                                       make_connection(from='protein', to='phosphosite',
                                        connect_on='gene_name', weight=1),
                                       make_connection(from='protein', to='metabolite',
                                        connect_on=metabolite_protein_interactions,
                                        weight='combined_score'))
```

---

make_drug_target *Reformat drug-target-interaction data*

Description

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

Usage

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

Arguments

target_molecules	[string] Name of layer containing the drug targets. This name has to match the corresponding named item in the list of layers supplied to run_pipeline.
interaction_table	[data.frame] Has to contain two columns. A column called 'drug_name' containing names or identifiers of drugs. And a column with a name that matches an identifier in the layer supplied in 'target_molecules'. Additional columns will be ignored in the pipeline. For example, if drugs target proteins and an identifier called 'ncbi_id' was supplied in layer creation of the protein layer (see 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 could also be used.
match_on	[string] 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 run_pipeline.

Examples

```
data(drug_gene_interactions)

example_drug_target_interactions <- 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 `run_pipeline`.

Usage

```
make_layer(
  name,
  data_groupA,
  data_groupB,
  identifiers_groupA,
  identifiers_groupB
)
```

Arguments

`name`	[string] Name of the layer.
`data_groupA`, `data_groupB`	[data.frame] Data frame containing raw molecular data of each group (each stratum). Analyzed components (e.g. genes) in columns, samples (e.g. patients) in rows.
`identifiers_groupA`, `identifiers_groupB`	[data.frame] 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 data set for one layer), that can be supplied to `run_pipeline` and ‘name‘ giving the name of the layer. Each sub-list contains the ‘data‘ and the ‘identifiers‘.

Examples

```
data(protein_data)

example_protein_layer <- make_layer(
                            name="protein",
                            data_groupA=protein_data$groupA[, c(-1,-2)],
                            data_groupB=protein_data$groupB[, c(-1,-2)],
                            identifiers_groupA=data.frame(
                                gene_name=protein_data$groupA$gene_name,
                                ref_seq=protein_data$groupA$ref_seq),
                            identifiers_groupB=data.frame(
                                gene_name=protein_data$groupB$gene_name,
                                ref_seq=protein_data$groupB$ref_seq))
```

---

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 (‘groupA‘ = ER+, ‘groupB‘ = ER-). The data was reduced to 50 metabolites. For each group a data frame is given containing the raw data with the metabolites as rows and the samples as columns. The first three columns contain the metabolite identifiers (biochemical_name, metabolon_id and pubchem_id).

Usage

```
metabolite_data
```

Format

groupA ER+ data; data.frame: first three columns contain metabolite identifiers biochemical_name, metabolon_id and pubchem_id; other columns are samples containing the quantified metabolite data per metabolite

groupB ER- data; data.frame: first three columns contain metabolite identifiers biochemical_name, metabolon_id and pubchem_id; other columns are samples containing the quantified metabolite data per metabolite

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_interactions *Metabolite protein interaction data*

Description

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

Usage

```
metabolite_protein_interactions
```

Format

A data frame with 3 columns.

pubchem_id Pubchem IDs defining interacting metabolites

gene_name gene names 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/

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 ('groupA' = ER+, 'groupB' = ER-). The data was reduced to 50 genes. For each group a data frame is given containing the raw data with the mRNA/gene as rows and the samples as columns. The first column contains the gene identifiers (gene_name).

Usage

```
mrna_data
```

Format

groupA ER+ data; data.frame: first column contains mRNA/gene identifier gene_name; other columns are samples containing the quantified mRNA data per gene

groupB ER- data; data.frame: first column contains mRNA/gene identifier gene_name; other columns are samples containing the quantified mRNA data per gene

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

---

phosphosite_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 ('groupA' = ER+, 'groupB' = ER-). The data was reduced to 50 genes. For each group a data frame is given containing the raw data with the phosphosites as rows and the samples as columns. The first three columns contain the phosphosite and protein identifiers (site_id, ref_seq and gene_name).

Usage

```
phosphosite_data
```

Format

groupA ER+ data; data.frame: first three columns contain phosphosite and protein identifiers site_id, ref_seq and gene_name; other columns are samples containing the quantified phosphosite data per phosphosite

groupB ER- data; data.frame: first three columns contain phosphosite and protein identifiers site_id, ref_seq and gene_name; other columns are samples containing the quantified phosphosite data per phosphosite

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

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 (‘groupA‘ = ER+, ‘groupB‘ = ER-). The data was reduced to 50 genes. For each group a data frame is given containing the raw data with the proteins as rows and the samples as columns. The first two columns contain the protein identifiers (ref_seq and gene_name).

Usage

```
protein_data
```

Format

groupA ER+ data; data.frame: first two columns contain protein identifiers ref_seq and gene_name; other columns are samples containing the quantified proteomics data per protein

groupB ER- data; data.frame: first two columns contain protein identifiers ref_seq and gene_name; other columns are samples containing the quantified proteomics data per protein

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 in the input data*

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	[string] Character string vector containing error messages.

Value

No return value, writes error messages to console

Examples

```
data(layers_example)
data(metabolite_protein_interactions)
data(drug_gene_interactions)
data

all_layers <- layers_example

all_inter_layer_connections = list(
    make_connection(from='mrna', to='protein', connect_on='gene_name', weight=1),
    make_connection(from='protein', to='phosphosite', connect_on='gene_name', weight=1),
    make_connection(from='protein', to='metabolite',
    connect_on=metabolite_protein_interactions, weight='combined_score'))

all_drug_target_interactions <- make_drug_target(
                                    target_molecules="protein",
                                    interaction_table=drug_gene_interactions,
                                    match_on="gene_name")

return_errors(check_input(layers=all_layers,
    inter_layer_connections=all_inter_layer_connections,
    drug_target_interactions=all_drug_target_interactions))
```

`run_pipeline` *Execute all DrDimont pipeline steps sequentially*

Description

This wrapper function executes all necessary steps to generate differential integrated 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.

As this function runs through all steps of the DrDimont pipeline it can take a long time to complete, especially if the supplied molecular data is rather large. Several prompts will be printed to supply information on how the pipeline is proceeding. Calculation of the interaction score by `generate_interaction_score_graphs` requires saving large-scale graphs to file and calling 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 DrDimont.

Usage

```
run_pipeline(
  layers,
  inter_layer_connections,
  drug_target_interactions,
  settings
)
```

Arguments

layers	[list] 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	[list] 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_interactions	[list] 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	[list] A named list containing pipeline settings. The settings list has to be initialized by drdimont_settings. Items in the named list can be adjusted as desired.

Value

Data frame containing drug name and associated differential integrated drug response 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(metabolite_protein_interactions)
data(layers_example)

example_inter_layer_connections = list(make_connection(from='mrna', to='protein',
                                       connect_on='gene_name', weight=1),
                                   make_connection(from='protein', to='phosphosite',
                                       connect_on='gene_name', weight=1),
                                   make_connection(from='protein', to='metabolite',
                                       connect_on=metabolite_protein_interactions,
                                       weight='combined_score'))

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

```
example_settings <- drdimont_settings(
                        handling_missing_data=list(
                            default="pairwise.complete.obs",
                            mrna="all.obs"),
                        reduction_method="pickHardThreshold",
                        r_squared=list(default=0.65, metabolite=0.1),
                        cut_vector=list(default=seq(0.2, 0.65, 0.01)))


run_pipeline(
    layers=layers_example,
    inter_layer_connections=example_inter_layer_connections,
    drug_target_interactions=example_drug_target_interactions,
    settings=example_settings)
```

Index