Package 'cosmosR'

April 9, 2025

Type Package

Title COSMOS (Causal Oriented Search of Multi-Omic Space)

Version 1.15.0

Description COSMOS (Causal Oriented Search of Multi-Omic Space) is a method that integrates phosphoproteomics, transcriptomics, and metabolomics data sets based on prior knowledge of signaling, metabolic, and gene regulatory networks. It estimated the activities of transcrption factors and kinases and finds a network-level causal reasoning. Thereby, COSMOS provides mechanistic hypotheses for experimental observations across mulit-omics datasets.

URL https://github.com/saezlab/COSMOSR

BugReports https://github.com/saezlab/COSMOSR/issues

Depends R (>= 4.1)

License GPL-3

Encoding UTF-8

LazyData false

RoxygenNote 7.2.3

VignetteBuilder knitr

Imports CARNIVAL, dorothea, dplyr, GSEABase, igraph, progress, purrr, rlang, stringr, utils, visNetwork, decoupleR

Suggests testthat, knitr, rmarkdown, piano, ggplot2

biocViews CellBiology, Pathways, Network, Proteomics, Metabolomics, Transcriptomics, GeneSignaling

git_url https://git.bioconductor.org/packages/cosmosR

git_branch devel

git_last_commit 7c6afd6

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2025-04-09

Author Aurélien Dugourd [aut] (ORCID: <https://orcid.org/0000-0002-0714-028X>),
Attila Gabor [cre] (ORCID: <https://orcid.org/0000-0002-0776-1182>),
Katharina Zirngibl [aut] (ORCID:
<https://orcid.org/0000-0002-7518-0339>)

Maintainer Attila Gabor <attila.gabor@uni-heidelberg.de>

Contents

`compress_same_children`

Compress Network by Merging Nodes with Identical Children

Description

This function compresses a network by merging nodes that have the same children. The input network is represented as a data frame with three columns: source, target, and sign of interaction. The function returns a list containing the compressed network, node signatures, and duplicated signatures.

Usage

```
compress_same_children(df, sig_input, metab_input)
```

Arguments

df	A data frame representing the network with three columns: source, target, and sign of interaction.
sig_input	A list of input node signatures to be considered for the merging process.
metab_input	A list of input metabolic signatures to be considered for the merging process.

Value

A list containing the following elements:

compressed_network
: A data frame representing the compressed network.

node_signatures
: A list of signatures of nodes in the network after the merging process.

duplicated_signatures
: A list of duplicated signatures in the network after the merging process.

Examples

```
# Create a sample network
df <- data.frame(source = c("A", "A", "B", "B"),
                 target = c("C", "D", "C", "D"),
                 sign_of_interaction = c(1, 1, 1, 1))

# Define input node and metabolic signatures
sig_input <- list()
metab_input <- list()

# Compress the network
result <- compress_same_children(df, sig_input, metab_input)
compressed_network <- result$compressed_network
```

---

cosmos_data *Create Cosmos Data*

Description

An S3 class that combines the required data into a comprehensive list. Use the preprocess_COSMOS_signaling_to_metabol or preprocess_COSMOS_metabolism_to_signaling to create an instance.

Usage

```
cosmos_data(
  meta_network,
  tf_regulon = NULL,
  signaling_data,
  metabolic_data,
  expression_data,
  verbose = TRUE
)
```

Arguments

meta_network	Prior knowledge network (PKN). By default COSMOS use a PKN derived from Omnipath, STITCHdb and Recon3D. See details on the data meta_network.
tf_regulon	Collection of transcription factor - target interactions. A default collection from dorothea can be obtained by the load_tf_regulon_dorothea function.
signaling_data	Numerical vector, where names are signaling nodes in the PKN and values are from {1, 0, -1}. Continuous data will be discretized using the sign function.
metabolic_data	Numerical vector, where names are metabolic nodes in the PKN and values are continuous values that represents log2 fold change or t-values from a differential analysis. These values are compared to the simulation results (simulated nodes can take value -1, 0 or 1).
expression_data	Numerical vector that represents the results of a differential gene expression analysis. Names are gene names using EntrezID starting with an X and values are log fold change or t-values. Genes with NA values are considered none expressed and they will be removed from the TF-gene expression interactions.
verbose	(default: TRUE) Reports details about the cosmos_data object.

Value

cosmos data class instance.

---

decompress_solution_network

Decompress Solution Network

Description

This function decompresses a solution network by mapping node signatures back to their original identifiers. The input is a formatted solution network, a meta network, node signatures, and duplicated parents. The function returns a list containing the decompressed solution network and attribute table.

Usage

```
decompress_solution_network(
  formatted_res,
  meta_network,
  node_signatures,
  duplicated_parents
)
```

Arguments

formatted_res	A list containing the solution network and attribute table.
meta_network	A data frame representing the meta network.
node_signatures	A list of node signatures.
duplicated_parents	A list of duplicated parents from the compression process.

Value

A list containing the following elements:

SIF	A data frame representing the decompressed solution network.
ATT	A data frame containing the attributes of the decompressed solution network.

Examples

```
# Create a sample formatted_res
formatted_res <- list(
  SIF = data.frame(source = c("parent_of_D1", "D"),
                   target = c("D", "F"),
                   interaction = c(1, 1),
                   Weight = c(1, 1)),
  ATT = data.frame(Nodes = c("parent_of_D1", "D", "F"),
                   NodeType = c("","",""),
                   ZeroAct = c(0,0,0),
                   UpAct = c(1,1,1),
                   DownAct = c(0,0,0),
                   AvgAct = c(1,1,1),
                   measured = c(0,0,0),
                   Activity = c(1,1,1))
)

# Create a sample meta_network
meta_network <- data.frame(source = c("A", "B", "D"),
                           target = c("D", "D", "F"),
                           interaction_type = c(1, 1, 1))

# Define node_signatures and duplicated_parents
node_signatures <- list("A" = "parent_of_D1","B" = "parent_of_D1","D" = "parent_F1")
duplicated_parents <- c("A" = "parent_of_D1","B" = "parent_of_D1")
```

```
# Decompress the solution network
result <- decompress_solution_network(formatted_res, meta_network, node_signatures, duplicated_parents)
decompressed_network <- result[[1]]
attribute_table <- result[[2]]
```

`decoupleRnival` *DecoupleRnival*

Description

Iteratively propagate downstream input activity through a signed directed network using the weighted mean enrichment score from decoupleR package

Usage

```
decoupleRnival(
  upstream_input = NULL,
  downstream_input,
  meta_network,
  n_layers,
  n_perm = 1000,
  downstream_cutoff = 0
)
```

Arguments

`upstream_input`	A named vector with up_stream nodes and their corresponding activity.
`downstream_input`	A named vector with down_stream nodes and their corresponding activity.
`meta_network`	A network data frame containing signed directed prior knowledge of molecular interactions.
`n_layers`	The number of layers that will be propagated upstream.
`n_perm`	The number of permutations to use in decoupleR's algorithm.
`downstream_cutoff`	If downstream measurments should be included above a given threshold

Value

A data frame containing the score of the nodes upstream of the downstream input based on the iterative propagation

Examples

```
# Example input data
upstream_input <- c("A" = 1, "B" = -1, "C" = 0.5)
downstream_input <- c("D" = 2, "E" = -1.5)
meta_network <- data.frame(
  source = c("A", "A", "B", "C", "C", "D", "E"),
  target = c("B", "C", "D", "E", "D", "B", "A"),
  sign = c(1, -1, -1, 1, -1, -1, 1)
)

# Run the function with the example input data
result <- decoupleRnival(upstream_input, downstream_input, meta_network, n_layers = 2, n_perm = 100)

# View the results
print(result)
```

`default_CARNIVAL_options` *Setting Default CARNIVAL Options*

Description

Returns the default CARNIVAL options as a list. You can modify the elements of the list and then use it as an argument in `run_COSMOS_metabolism_to_signaling` or `run_COSMOS_signaling_to_metabolism`. If you choose CPLEX or CBC, you must modify then the solverPath field and point to the CPLEX/CBC executable (See Details).

Usage

```
default_CARNIVAL_options(solver = NULL)
```

Arguments

`solver`	one of 'cplex' (recommended, but require 3rd party tool), 'cbc' (also require 3rd party tool) or 'lpSolve' (only for small networks)

Details

COSMOS is dependent on CARNIVAL for exhibiting the signalling pathway optimisation. CARNIVAL requires the interactive version of IBM Cplex, Gurobi or CBC-COIN solver as the network optimiser. The IBM ILOG Cplex is freely available through Academic Initiative here. Gurobi license is also free for academics, request a license following instructions here. The CBC solver is open source and freely available for any user, but has a significantly lower performance than CPLEX or Gurobi. Obtain CBC executable directly usable for cosmos here. Alternatively for small networks, users can rely on the freely available lpSolve R-package, which is automatically installed with the package.

Value

returns a list with all possible options implemented in CARNIVAL. see the documentation on runCARNIVAL.

Examples

```
# load and change default options:
my_options = default_CARNIVAL_options(solver = "cplex")

my_options$solverPath = "/Applications/CPLEX_Studio128/cplex/bin/x86-64_osx/cplex"
my_options$threads = 2
my_options$timelimit = 3600*15
```

---

display_node_neighboorhood

display_node_neighboorhood

Description

display input and measurements within n steps of a given set of nodes

Usage

```
display_node_neighboorhood(central_node, sif, att, n = 100)
```

Arguments

central_node	character or character vector; node ID(s) around which a network will be branched out untill meansurments and input are reached
sif	df; COSMOS network solution in sif format like the first list element returned by the format_cosmos_res function
att	df; attributes of the nodes of the COMSOS network solution like the second list element returned by the format_cosmos_res function
n	numeric; maximum number of steps in the network to look for inputs and measurments

Value

a visnetwork object

Examples

```
CARNIVAL_options <- cosmosR::default_CARNIVAL_options("lpSolve")
data(toy_network)
data(toy_signaling_input)
data(toy_metabolic_input)
data(toy_RNA)
test_for <- preprocess_COSMOS_signaling_to_metabolism(meta_network = toy_network,
signaling_data = toy_signaling_input,
metabolic_data = toy_metabolic_input,
diff_expression_data = toy_RNA,
maximum_network_depth = 15,
remove_unexpressed_nodes = TRUE,
CARNIVAL_options = CARNIVAL_options
)
test_result_for <- run_COSMOS_signaling_to_metabolism(data = test_for,
CARNIVAL_options = CARNIVAL_options)
test_result_for <- format_COSMOS_res(test_result_for)
network_plot <- display_node_neighboorhood(central_node = 'MYC',
sif = test_result_for[[1]],
att = test_result_for[[2]],
n = 7)
network_plot
```

---

extract_nodes_for_ORA *Extract COSMOS nodes for ORA analysis*

Description

Function to extract the nodes that appear in the COSMOS output network and the background genes (all genes present in the prior knowledge network)

Usage

```
extract_nodes_for_ORA(sif, att)
```

Arguments

sif	df; COSMOS network solution in sif format like the first list element returned by the format_cosmos_res function
att	df; attributes of the nodes of the COMSOS network solution like the second list element returned by the format_cosmos_res function

Value

List with 2 objects: the success and the background genes

Examples

```
CARNIVAL_options <- cosmosR::default_CARNIVAL_options("lpSolve")
data(toy_network)
data(toy_signaling_input)
data(toy_metabolic_input)
data(toy_RNA)
test_for <- preprocess_COSMOS_signaling_to_metabolism(meta_network = toy_network,
signaling_data = toy_signaling_input,
metabolic_data = toy_metabolic_input,
diff_expression_data = toy_RNA,
maximum_network_depth = 15,
remove_unexpressed_nodes = TRUE,
CARNIVAL_options = CARNIVAL_options
)
test_result_for <- run_COSMOS_signaling_to_metabolism(data = test_for,
CARNIVAL_options = CARNIVAL_options)
test_result_for <- format_COSMOS_res(test_result_for)
extreacted_nodes <- extract_nodes_for_ORA(
sif = test_result_for[[1]],
att = test_result_for[[2]]
)
```

`filter_incohrent_TF_target` *filter_incohrent_TF_target*

Description

Filters incoherent target genes from a regulatory network based on a decoupling analysis of upstream and downstream gene expression.

Usage

```
filter_incohrent_TF_target(
  decouplRnival_res,
  TF_reg_net,
  meta_network,
  RNA_input
)
```

Arguments

`decouplRnival_res`	A data frame resulting from the decoupleRnival function.
`TF_reg_net`	A data frame containing prior knowledge of transcription factor (TF) regulatory interactions.
`meta_network`	A network data frame containing signed directed prior knowledge of molecular interactions.
`RNA_input`	A named vector containing differential gene expression data.

Value

A network data frame containing the genes that are not incoherently regulated by TFs.

Examples

```
# Example input data
upstream_input <- c("A" = 1, "B" = -1, "C" = 0.5)
downstream_input <- c("D" = 2, "E" = -1.5)
meta_network <- data.frame(
  source = c("A", "A", "B", "C", "C", "D", "E"),
  target = c("B", "D", "D", "E", "D", "B", "A"),
  interaction = c(-1, 1, -1, 1, -1, -1, 1)
)
RNA_input <- c("A" = 1, "B" = -1, "C" = 5, "D" = -0.7, "E" = -0.3)

TF_reg_net <- data.frame(
source = c("B"),
target = c("D"),
mor = c(-1)
)

# Run the decoupleRnival function to get the upstream influence scores
upstream_scores <- decoupleRnival(upstream_input, downstream_input, meta_network, n_layers = 2, n_perm = 100)

filtered_network <- filter_incohrent_TF_target(upstream_scores, TF_reg_net, meta_network, RNA_input)

print(filtered_network)
```

---

format_COSMOS_res *format_COSMOS_res*

Description

formats the network with readable names

Usage

```
format_COSMOS_res(cosmos_res, metab_mapping = NULL)
```

Arguments

cosmos_res	results of COSMOS run
metab_mapping	a named vector with HMDB Ids as names and desired metabolite names as values.

Value

list with network and attribute tables.

gmt_to_dataframe *Convert gmt file to data frame*

Description

This function is designed to convert a gmt file (gene set file from MSigDB) into a two column data frame where the first column corresponds to omic features (genes) and the second column to associated terms (pathway the gene belongs to). One gene can belong to several pathways.

Usage

```
gmt_to_dataframe(gmtfile)
```

Arguments

gmtfile	a full path name of the gmt file to be converted

Value

a two column data frame where the first column corresponds to omic features and the second column to associated terms (pathways).

HMDB_mapper_vec *Toy Input Transcription Data Set*

Description

This exemplary transcription data are the specific deregulated gene expression of the 786-O cell line from the NCI60 dataset.

Usage

```
data(HMDB_mapper_vec)
```

Format

An object of class "character" containing the marching between HMDB metabolite IDs and there coresponding metabolite names.

Source

https://bioconductor.org/packages/release/data/annotation/html/metaboliteIDmapping.html

Examples

```
data(HMDB_mapper_vec)
```

load_tf_regulon_dorothea *load transcription factor regulon*

Description

load the transcription factors from DOROTHEA package and converts gene symbols to EntrezID using org.Hs.eg.db

Usage

```
load_tf_regulon_dorothea(confidence = c("A", "B", "C"))
```

Arguments

confidence	strong vector (by default: c("A","B","C")). Subset of {A, B, C, D, E}. See the 'dorothea' for the meaning of confidence levels. package for further details.

Value

returns a PKN of a form of a data table. Each row is an interaction. Columns names are:

- 'tf' transcription factor - 'confidence' class of confidence - 'target' target gene - 'sign' indicates if interaction is up (1) or down-regulation (-1).

Examples

```
load_tf_regulon_dorothea()
```

meta_network *Meta Prior Knowledge Network*

Description

Comprehensive Prior Knowledge Network (PKN), which combines signaling and metabolic interaction networks. The network was constructed using the Recon3D and STITCH metabolic networks as well as the signaling network from OmniPath.

Usage

```
data(meta_network)
```

Format

An object of class "tibble" with 117065 rows (interactions) and three variables:

`source` Source node, either metabolite or protein

`interaction` Type of interaction, 1 = Activation, -1 = Inhibition

`target` Target node, either metabolite or protein A detailed description of the identifier formatting can be found under https://metapkn.omnipathdb.org/00__README.txt.

Source

The network is available in Omnipath: https://metapkn.omnipathdb.org/metapkn__20200122.txt, the scripts used for the build of the network are available under https://github.com/saezlab/Meta_PKN.

References

Dugourd, A., Kuppe, C. and Sciacovelli, M. et. al. (2021) *Molecular Systems Biology*. **17**, e9730.

Examples

```
data(meta_network)
```

`meta_network_cleanup` *meta_network_cleanup*

Description

This function cleans up a meta network data frame by removing self-interactions, calculating the mean interaction values for duplicated source-target pairs, and keeping only interactions with values of 1 or -1.

Usage

```
meta_network_cleanup(meta_network)
```

Arguments

`meta_network` A data frame with columns 'source', 'interaction', and 'target'.

Value

A cleaned up meta network data frame.

Examples

```
# Create a meta network data frame
example_meta_network <- data.frame(
source = c("A", "B", "C", "D", "A", "B", "C", "D", "A"),
interaction = c(1, 1, 1, -1, 1, -1, 1, -1, 1),
target = c("B", "C", "D", "A", "C", "D", "A", "B", "B")
)

# Clean up the example meta network
cleaned_meta_network <- meta_network_cleanup(example_meta_network)
print(cleaned_meta_network)
```

`prepare_metab_inputs` *add metabolic compartment and metab__ prefix to metabolite IDs*

Description

This function adds metabolic compartments to the metabolic identifiers provided by the user.

Usage

```
prepare_metab_inputs(metab_input, compartment_codes)
```

Arguments

`metab_input`	a named vector with matebolic statistics as inputs and metabolite identifiers as names
`compartment_codes`	a character vector, the desired compartment codes to be added. Possible values are "r", "c", "e", "x", "m", "l", "n" and "g"

Value

a named vector with the compartment code and prefixed added to the names

---

`preprocess_COSMOS_metabolism_to_signaling`
Preprocess COSMOS Inputs For Metabolism to Signaling

Description

Runs checks on the input data and simplifies the prior knowledge network. Simplification includes the removal of (1) nodes that are not reachable from signaling nodes and (2) interactions between transcription factors and target genes if the target gene does not respond or the response is contradictory with the change in the transcription factor activity. Optionally, further TF activities are estimated via network optimization via CARNIVAL and the interactions between TF and genes are filtered again.

Usage

```
preprocess_COSMOS_metabolism_to_signaling(
  meta_network = meta_network,
  tf_regulon = load_tf_regulon_dorothea(),
  signaling_data,
  metabolic_data,
  diff_expression_data = NULL,
  diff_exp_threshold = 1,
  maximum_network_depth = 8,
  expressed_genes = NULL,
  remove_unexpressed_nodes = TRUE,
  filter_tf_gene_interaction_by_optimization = TRUE,
  CARNIVAL_options = default_CARNIVAL_options("lpSolve")
)
```

Arguments

meta_network	prior knowledge network (PKN). A PKN released with COSMOS and derived from Omnipath, STITCHdb and Recon3D can be used. See details on the data meta_network.
tf_regulon	collection of transcription factor - target interactions. A default collection from dorothea can be obtained by the load_tf_regulon_dorothea function.
signaling_data	numerical vector, where names are signaling nodes in the PKN and values are from {1, 0, -1}. Continuous data will be discretized using the sign function.
metabolic_data	numerical vector, where names are metabolic nodes in the PKN and values are continuous values that represents log2 fold change or t-values from a differential analysis. These values are compared to the simulation results (simulated nodes can take value -1, 0 or 1)
diff_expression_data	(optional) numerical vector that represents the results of a differential gene expression analysis. Names are gene names using gene symbole and values are log fold change or t-values. We use the "diff_exp_threshold" parameter to decide which genes changed significantly. Genes with NA values are considered none expressed and they will be removed from the TF-gene expression interactions.
diff_exp_threshold	threshold parameter (default 1) used to binarize the values of "diff_expression_data".
maximum_network_depth	integer > 0 (default: 8). Nodes that are further than "maximum_network_depth" steps from the signaling nodes on the directed graph of the PKN are considered non-reachable and are removed.
expressed_genes	character vector. Names of nodes that are expressed. By default we consider all the nodes that appear in diff_expression_data with a numeric value (i.e. nodes with NA are removed)
remove_unexpressed_nodes	if TRUE (default) removes nodes from the PKN that are not expressed, see input "expressed_genes".

filter_tf_gene_interaction_by_optimization
: (default:TRUE), if TRUE then runs a network optimization that estimates TF activity not included in the inputs and checks the consistency between the estimated activity and change in gene expression. Removes interactions where TF and gene expression are inconsistent

CARNIVAL_options
: list that controls the options of CARNIVAL. See details in default_CARNIVAL_options.

Value

cosmos_data object with the following fields:

meta_network Filtered PKN

tf_regulon TF - target regulatory network

signaling_data_bin Binarised signaling data

metabolic_data Metabolomics data

diff_expression_data_bin Binarized gene expression data

optimized_network Initial optimized network if filter_tf_gene_interaction_by_optimization is TRUE

See Also

meta_network for meta PKN, load_tf_regulon_dorothea for tf regulon, runCARNIVAL.

Examples

```
data(toy_network)
data(toy_signaling_input)
data(toy_metabolic_input)
data(toy_RNA)
test_back <- preprocess_COSMOS_metabolism_to_signaling(
meta_network = toy_network,
signaling_data = toy_signaling_input,
metabolic_data = toy_metabolic_input,
diff_expression_data = toy_RNA,
maximum_network_depth = 15,
remove_unexpressed_nodes = TRUE,
CARNIVAL_options = default_CARNIVAL_options("lpSolve")
)
```

---

preprocess_COSMOS_signaling_to_metabolism

Preprocess COSMOS Inputs For Signaling to Metabolism

Description

Runs checks on the input data and simplifies the prior knowledge network. Simplification includes the removal of (1) nodes that are not reachable from signaling nodes and (2) interactions between transcription factors and target genes if the target gene does not respond or the response is contradictory with the change in the transcription factor activity. Optionally, further TF activities are estimated via network optimization via CARNIVAL and the interactions between TF and genes are filtered again.

Usage

```
preprocess_COSMOS_signaling_to_metabolism(
  meta_network = meta_network,
  tf_regulon = load_tf_regulon_dorothea(),
  signaling_data,
  metabolic_data,
  diff_expression_data = NULL,
  diff_exp_threshold = 1,
  maximum_network_depth = 8,
  expressed_genes = NULL,
  remove_unexpressed_nodes = TRUE,
  filter_tf_gene_interaction_by_optimization = TRUE,
  CARNIVAL_options = default_CARNIVAL_options("lpSolve")
)
```

Arguments

`meta_network`	prior knowledge network (PKN). A PKN released with COSMOS and derived from Omnipath, STITCHdb and Recon3D can be used. See details on the data `meta_network`.
`tf_regulon`	collection of transcription factor - target interactions. A default collection from dorothea can be obtained by the `load_tf_regulon_dorothea` function.
`signaling_data`	numerical vector, where names are signaling nodes in the PKN and values are from {1, 0, -1}. Continuous data will be discretized using the `sign` function.
`metabolic_data`	numerical vector, where names are metabolic nodes in the PKN and values are continuous values that represents log2 fold change or t-values from a differential analysis. These values are compared to the simulation results (simulated nodes can take value -1, 0 or 1)
`diff_expression_data`	(optional) numerical vector that represents the results of a differential gene expression analysis. Names are gene names using gene symbole and values are log fold change or t-values. We use the "`diff_exp_threshold`" parameter to decide which genes changed significantly. Genes with NA values are considered none expressed and they will be removed from the TF-gene expression interactions.
`diff_exp_threshold`	threshold parameter (default 1) used to binarize the values of "`diff_expression_data`".

`maximum_network_depth`
: integer > 0 (default: 8). Nodes that are further than "`maximum_network_depth`" steps from the signaling nodes on the directed graph of the PKN are considered non-reachable and are removed.

`expressed_genes`
: character vector. Names of nodes that are expressed. By default we consider all the nodes that appear in `diff_expression_data` with a numeric value (i.e. nodes with NA are removed)

`remove_unexpressed_nodes`
: if TRUE (default) removes nodes from the PKN that are not expressed, see input "`expressed_genes`".

`filter_tf_gene_interaction_by_optimization`
: (default:TRUE), if TRUE then runs a network optimization that estimates TF activity not included in the inputs and checks the consistency between the estimated activity and change in gene expression. Removes interactions where TF and gene expression are inconsistent

`CARNIVAL_options`
: list that controls the options of CARNIVAL. See details in `default_CARNIVAL_options`.

Value

cosmos_data object with the following fields:

`meta_network` Filtered PKN

`tf_regulon` TF - target regulatory network

`signaling_data_bin` Binarised signaling data

`metabolic_data` Metabolomics data

`diff_expression_data_bin` Binarized gene expression data

`optimized_network` Initial optimized network if `filter_tf_gene_interaction_by_optimization` is `TRUE`

See Also

`meta_network` for meta PKN, `load_tf_regulon_dorothea` for tf regulon, `runCARNIVAL`.

Examples

```
data(toy_network)
data(toy_signaling_input)
data(toy_metabolic_input)
data(toy_RNA)
test_for <- preprocess_COSMOS_signaling_to_metabolism(meta_network = toy_network,
     signaling_data = toy_signaling_input,
     metabolic_data = toy_metabolic_input,
     diff_expression_data = toy_RNA,
     maximum_network_depth = 15,
     remove_unexpressed_nodes = TRUE,
     CARNIVAL_options = default_CARNIVAL_options("lpSolve"))
```

print.cosmos_data *Print Cosmos Data Summary Print a summary of cosmos data.*

Description

Print Cosmos Data Summary Print a summary of cosmos data.

Usage

```
## S3 method for class 'cosmos_data'
print(x, ...)
```

Arguments

x	cosmos_data object. Use the preprocess_COSMOS_signaling_to_metabolism or preprocess_COSMOS_metabolism_to_signaling functions to create one.
...	Further print arguments passed to or from other methods.

Value

input (invisible)

See Also

print, cosmos_data

reduce_solution_network *reduce_solution_network*

Description

Reduces a solution network based on a decoupling analysis of upstream and downstream gene expression, by filtering out edges that do not meet a consistency threshold, and limiting the network to a certain number of steps from upstream input nodes.

Usage

```
reduce_solution_network(
  decoupleRnival_res,
  meta_network,
  cutoff,
  upstream_input,
  RNA_input = NULL,
  n_steps = 10
)
```

Arguments

decoupleRnival_res	A data frame resulting from the decoupleRnival function.
meta_network	A network data frame containing signed directed prior knowledge of molecular interactions.
cutoff	The consistency threshold for filtering edges from the solution network.
upstream_input	A named vector with up_stream nodes and their corresponding activity.
RNA_input	A named vector containing differential gene expression data.
n_steps	The maximum number of steps from upstream input nodes to include in the solution network.

Value

A list containing the solution network (SIF) and an attribute table (ATT) with gene expression data.

Examples

```
# Example input data
upstream_input <- c("A" = 1, "B" = -1, "C" = 0.5)
downstream_input <- c("D" = 2, "E" = -1.5)
meta_network <- data.frame(
  source = c("A", "A", "B", "C", "C", "D", "E"),
  target = c("B", "D", "D", "E", "D", "B", "A"),
  interaction = c(-1, 1, -1, 1, -1, -1, 1)
)
RNA_input <- c("A" = 1, "B" = -1, "C" = 5, "D" = 0.7, "E" = -0.3)

# Run the decoupleRnival function to get the upstream influence scores
upstream_scores <- decoupleRnival(upstream_input, downstream_input, meta_network, n_layers = 2, n_perm = 100)

# Reduce the solution network based on the upstream influence scores
reduced_network <- reduce_solution_network(upstream_scores, meta_network, 0.4, upstream_input, RNA_input, 3)

# View the resulting solution network and attribute table
print(reduced_network$SIF)
print(reduced_network$ATT)
```

---

run_COSMOS_metabolism_to_signaling

run COSMOS metabolism to signaling

Description

Runs COSMOS from metabolism to signaling. This function uses CARNIVAL to find a subset of the prior knowledge network based on optimization that (1) includes the most measured and input nodes and (2) which is in agreement with the data. Use preprocess_COSMOS_metabolism_to_signaling to prepare the the inputs, measurements and the prior knowledge network.

Usage

```
run_COSMOS_metabolism_to_signaling(
  data,
  CARNIVAL_options = default_CARNIVAL_options("lpSolve")
)
```

Arguments

data	cosmos_data object. Use the preprocess_COSMOS_metabolism_to_signaling function to create an instance.
CARNIVAL_options	List that controls the options of CARNIVAL. See details in default_CARNIVAL_options.

Value

List with the following elements:

`weightedSIF` The averaged networks found by optimization in a format of a Simple Interaction network, i.e. each row codes an edge

`N_networks` Number of solutions found by the optimization

`nodesAttributes` Estimated node properties

`individual_networks` List of optimial networks found

`individual_networks_node_attributes` Node activity in each network

See Also

preprocess_COSMOS_metabolism_to_signaling, runCARNIVAL, cosmos_data

Examples

```
data(toy_network)
data(toy_signaling_input)
data(toy_metabolic_input)
data(toy_RNA)
test_back <- preprocess_COSMOS_metabolism_to_signaling(meta_network = toy_network,
                        signaling_data = toy_signaling_input,
                        metabolic_data = toy_metabolic_input,
                        diff_expression_data = toy_RNA,
                        maximum_network_depth = 15,
                        remove_unexpressed_nodes = TRUE,
                        CARNIVAL_options = default_CARNIVAL_options("lpSolve"))

test_result_back <- run_COSMOS_metabolism_to_signaling(data = test_back,
                      CARNIVAL_options = default_CARNIVAL_options("lpSolve"))
```

run_COSMOS_signaling_to_metabolism *run COSMOS signaling to metabolism*

Description

Runs COSMOS from signaling to metabolism. This function uses CARNIVAL to find a subset of the prior knowledge network based on optimisation that (1) includes the most measured and input nodes and (2) which is in agreement with the data. Use preprocess_COSMOS_signaling_to_metabolism to prepare inputs, measurements and prior knowledge network.

Usage

```
run_COSMOS_signaling_to_metabolism(
  data,
  CARNIVAL_options = default_CARNIVAL_options("lpSolve")
)
```

Arguments

data	cosmos_data object. Use the preprocess_COSMOS_signaling_to_metabolism function to create an instance.
CARNIVAL_options	List that controls the options of CARNIVAL. See details in default_CARNIVAL_options.

Value

List with the following elements:

weightedSIF The averaged networks found by optimization in a format of a Simple Interaction network, i.e. each row codes an edge

N_networks Number of solutions found by the optimization

nodesAttributes Estimated node properties

individual_networks List of optimial networks found

individual_networks_node_attributes Node activity in each network

See Also

preprocess_COSMOS_metabolism_to_signaling, runCARNIVAL, cosmos_data

Examples

```
data(toy_network)
data(toy_signaling_input)
data(toy_metabolic_input)
data(toy_RNA)
test_for <- preprocess_COSMOS_signaling_to_metabolism(meta_network = toy_network,
```

```
                              signaling_data = toy_signaling_input,
                              metabolic_data = toy_metabolic_input,
                              diff_expression_data = toy_RNA,
                              maximum_network_depth = 15,
                              remove_unexpressed_nodes = TRUE,
                              CARNIVAL_options = default_CARNIVAL_options("lpSolve"))

test_result_for <- run_COSMOS_signaling_to_metabolism(data = test_for,
                              CARNIVAL_options = default_CARNIVAL_options("lpSolve"))
```

`toy_metabolic_input` *Toy Metabolic Input Data*

Description

This metabolic data are a subset from the metabolic measurements of the 786-O cell line from the NCI60 dataset.

Usage

```
data(toy_metabolic_input)
```

Format

An object of class "`numeric`" containing the t-values of 2 metabolites, which are named with metabolite HMDB Ids matching the toy network.

Source

Subset of: https://github.com/saezlab/COSMOS_MSB/blob/main/data/metab_input_COSMOS.csv

References

Dugourd, A., Kuppe, C. and Sciacovelli, M. et. al. (2021) *Molecular Systems Biology*. **17**, e9730.

Examples

```
data(toy_metabolic_input)
```

toy_network *Toy Input Network*

Description

This signaling network is the reduced COSMOS network solution obtained in the cosmos test on 786-O NCI60 data. Here, this network solution is reused as an exemplary input prior knowledge network (PKN).

Usage

```
data(toy_network)
```

Format

An object of class "data.frame" with 19 rows (interactions) and three variables:

`source` Source node, either metabolite or protein

`interaction` Type of interaction, 1 = Activation, -1 = Inhibition

`target` Target node, either metabolite or protein A detailed description of the identifier formatting can be found under https://metapkn.omnipathdb.org/00__README.txt.

Source

The network data are available on github: https://github.com/saezlab/COSMOS_MSB/tree/main/results/COSMOS_result/COSMOS_res_session.RData. The toy_network is the combined network of the COSMOS network solutions CARNIVAL_Result2 and CARNIVAL_Result_rerun subsequently reduced to 19 exemplary nodes.

References

Dugourd, A., Kuppe, C. and Sciacovelli, M. et. al. (2021) *Molecular Systems Biology*. **17**, e9730.

Examples

```
data(toy_network)
```

toy_RNA *Toy Input Transcription Data Set*

Description

This exemplary transcription data are the specific deregulated gene expression of the 786-O cell line from the NCI60 dataset.

Usage

```
data(toy_RNA)
```

Format

An object of class "numeric" containing the t-values of 9300 genes, which are named with gene symboles matching the toy network.

Source

https://github.com/saezlab/COSMOS_MSB/blob/main/data/RNA_ttop_tumorvshealthy.csv

References

Dugourd, A., Kuppe, C. and Sciacovelli, M. et. al. (2021) *Molecular Systems Biology*. **17**, e9730.

Examples

```
data(toy_RNA)
```

toy_signaling_input *Toy Signaling Input*

Description

This signaling data are a subset of the footprint-based signaling activity estimates of transcription factors of the 786-O cell line from the NCI60 dataset.

Usage

```
data(toy_signaling_input)
```

Format

An object of class "data.frame" containing the normalised enrichment scores (NES) of 2 signaling proteins, which are named with their respective gene Entrez ID matching the toy network.

Source

Subset of: https://github.com/saezlab/COSMOS_MSB/blob/main/data/signaling_input_COSMOS.csv

References

Dugourd, A., Kuppe, C. and Sciacovelli, M. et. al. (2021) *Molecular Systems Biology*. **17**, e9730.

Examples

```
data(toy_signaling_input)
```

Index