Package 'cosmosR'

November 11, 2026

Type Package

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

Version 1.20.0

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 7.3.2

VignetteBuilder knitr

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

Suggests testthat, knitr, rmarkdown, htmltools, markdown, ggplot2, stringi, reshape2

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

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

git_branch RELEASE_3_23

git_last_commit 3fdc39f

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

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

Maintainer Aurélien Dugourd <dugourd@ebi.ac.uk>

Contents

compress_same_children . . . 3
cosmos_data . . . 4
createLinearColors . . . 5
decompress_moon_result . . . 6
decompress_solution_network . . . 6
decoupleRnival . . . 8
default_CARNIVAL_options . . . 9
display_node_neighboorhood . . . 10
extract_nodes_for_ORA . . . 11
filter_incohrent_TF_target . . . 12
format_COSMOS_res . . . 13
format_LR_ressource . . . 14
get_moon_scoring_network . . . 15
gmt_to_dataframe . . . 16
HMDB_mapper_vec . . . 16
load_tf_regulon_dorothea . . . 17
make_heatmap_color_palette . . . 17
meta_network . . . 18
meta_network_cleanup . . . 19
moon . . . 19
prepare_metab_inputs . . . 21
preprocess_COSMOS_metabolism_to_signaling . . . 21
preprocess_COSMOS_signaling_to_metabolism . . . 23
print.cosmos_data . . . 25
reduce_solution_network . . . 26
reduce_solution_network_double_thresh . . . 27
run_COSMOS_metabolism_to_signaling . . . 29
run_COSMOS_signaling_to_metabolism . . . 30
toy_metabolic_input . . . 31
toy_network . . . 32
toy_RNA . . . 33
toy_signaling_input . . . 33
translate_column_HMDB . . . 34
translate_res . . . 35
wide_ulm_res . . . 36

Index **37**

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.

createLinearColors *Create Linear Colors Based on Numeric Input*

Description

This function generates a gradient of colors based on the provided numeric values. The colors can be adjusted to include zero and are configurable with a specified maximum and custom color palette.

Usage

```
createLinearColors(
  numbers,
  withZero = T,
  maximum = 100,
  my_colors = c("royalblue3", "white", "red")
)
```

Arguments

numbers	A numeric vector for which the color gradient is to be generated.
withZero	A logical value indicating whether zero should be included in the color gradient. Default is TRUE.
maximum	An integer specifying the maximum number of colors to be generated in the gradient. Default is 100.
my_colors	A character vector of length three specifying the colors to be used in the gradient. Default is c("royalblue3", "white", "red").

Value

A character vector of colors representing the gradient based on the input numeric values.

Examples

```
# Generate colors for a set of numbers including zero
numbers <- c(-50, -20, 0, 20, 50)
colors <- createLinearColors(numbers, withZero = TRUE, maximum = 100)
```

decompress_moon_result

Decompress Moon Result

Description

This function decompresses the results obtained from moon analysis by incorporating node signatures and handling duplicated parents. It merges these details with the provided meta network data and returns a comprehensive data frame.

Usage

```
decompress_moon_result(moon_res, meta_network_compressed_list, meta_network)
```

Arguments

moon_res	A data frame containing the results of a moon analysis.
meta_network_compressed_list	A list containing compressed meta network details, including node signatures and duplicated parents.
meta_network	A data frame representing the original meta network.

Value

A data frame which merges the moon analysis results with the meta network data, including additional details about node signatures and handling of duplicated parents.

Examples

```
# Example usage (requires appropriate data structures for moon_res,
# meta_network_compressed_list, and meta_network)
# decompressed_result <- decompress_moon_result(moon_res, meta_network_compressed_list, meta_network)
```

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,
  statistic = "norm_wmean"
)
```

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
`statistic`	the decoupleR stat to consider: "wmean", "norm_wmean", or "ulm"

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.

`format_LR_ressource` *Format Ligand-Receptor Resource*

Description

This function formats a ligand-receptor resource by creating a gene set with source-target pairs, converting it to a long format, and adding default values for 'mor' and 'likelihood'.

Usage

```
format_LR_ressource(ligrec_ressource)
```

Arguments

`ligrec_ressource`	A data frame representing the ligand-receptor resource with columns for source and target gene symbols.

Value

A data frame containing the formatted ligand-receptor gene set with columns:

`gene`	The gene symbol from the ligand-receptor pairs.
`set`	The set identifier combining source and target gene symbols.
`mor`	Default value set to 1 for all entries.
`likelihood`	Default value set to 1 for all entries.

Examples

```
# Create a sample ligand-receptor resource
ligrec_ressource <- data.frame(source_genesymbol = c("L1", "L2"),
                               target_genesymbol = c("R1", "R2"))

# Format the ligand-receptor resource
formatted_geneset <- format_LR_ressource(ligrec_ressource)
```

get_moon_scoring_network *Get Moon Scoring Network*

Description

This function analyzes a given meta network based on moon scores and an upstream node. It filters and processes the network by controlling and observing neighbours according to specified parameters. The function returns a list containing a filtered network and updated moon scores.

Usage

```
get_moon_scoring_network(
  upstream_node,
  meta_network,
  moon_scores,
  keep_upstream_node_peers = F
)
```

Arguments

upstream_node	The node from which the network analysis starts.
meta_network	The complete network data.
moon_scores	Scores associated with each node in the network.
keep_upstream_node_peers	Logical; whether to keep peers of the upstream node. Default is FALSE.

Value

A list with two elements: - 'SIF': A data frame representing the filtered meta network. - 'ATT': A data frame representing the updated moon scores.

Examples

```
# Example usage (requires appropriate data structures for meta_network and moon_scores)
# result <- get_moon_scoring_network(upstream_node, meta_network, moon_scores)
```

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()
```

make_heatmap_color_palette

Create Heatmap Color Palette

Description

This function generates a color palette suitable for heatmaps based on the values in a matrix. It uses the 'createLinearColors' function to generate separate color gradients for positive and negative values.

Usage

```
make_heatmap_color_palette(my_matrix)
```

Arguments

my_matrix	A numeric matrix for which the heatmap color palette is to be generated.

Value

A character vector of colors representing the heatmap color palette based on the input matrix values.

Examples

```
# Create a sample matrix
my_matrix <- matrix(c(-3, -1, 0, 1, 3), nrow = 1)

# Generate heatmap color palette
heatmap_palette <- make_heatmap_color_palette(my_matrix)
```

---

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)
```

moon *moon*

Description

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

Usage

```
moon(
  upstream_input = NULL,
  downstream_input,
  meta_network,
  n_layers,
  n_perm = 1000,
  downstream_cutoff = 0,
  statistic = "ulm",
  return_levels = F
)
```

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
statistic	the decoupleR stat to consider: "wmean", "norm_wmean", or "ulm"
return_levels	true or false, if true the layers that the protein belongs to will be returned alongside the scores

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 <- moon(upstream_input, downstream_input, meta_network, n_layers = 2, statistic = "wmean")

# View the results
print(result)
```

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

Extracts a subnetwork from a decoupleRnival result and a prior knowledge network, enforcing score thresholds, neighbourhood distance, and structural constraints: - Pure children (no outgoing edges) must be level 0 in decoupleRnival_res. - Pure parents (no incoming edges) must be among the provided upstream_input nodes. - All nodes must have |score| > cutoff.

Usage

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

Arguments

decoupleRnival_res	A data.frame with columns 'source', 'score', and 'level' from decoupleRnival().
meta_network	A data.frame with columns 'source', 'target', 'interaction' for signed directed edges.

cutoff	Numeric. Absolute score threshold for node filtering.
upstream_input	A named numeric vector of upstream seed nodes and their activity values.
RNA_input	Optional named numeric vector of differential expression values; merged into ATT.
n_steps	Integer. Maximum number of steps away from any upstream_input node.

Value

A list with: - SIF: data.frame of filtered edges (‘source‘, ‘target‘, ‘interaction‘, ‘consistency‘). - ATT: data.frame of node attributes (‘nodes‘, ‘score‘, ‘level‘, ‘RNA_input‘).

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 decoupleRnival to generate scores
dec_res <- moon(upstream_input, downstream_input, meta_network,
                         n_layers = 2, n_perm = 100)
# Extract solution network
sol_net <- reduce_solution_network(dec_res, meta_network,
                                   cutoff = 0.4, upstream_input,
                                   RNA_input, n_steps = 3)
# View outputs
print(sol_net$SIF)
print(sol_net$ATT)
```

---

reduce_solution_network_double_thresh

reduce_solution_network_double_thresh

Description

Extracts a subnetwork using a two-tier absolute-score threshold, then restricts to only paths connecting upstream_input seeds to level 0 nodes, with recursive consistency filtering to ensure edge signs match node activities.

Usage

```
reduce_solution_network_double_thresh(
  decoupleRnival_res,
  meta_network,
```

```
  primary_thresh,
  secondary_thresh,
  upstream_input,
  RNA_input = NULL
)
```

Arguments

decoupleRnival_res	A data.frame with columns ‘source‘, numeric ‘score‘, and integer ‘level‘.
meta_network	A data.frame with columns ‘source‘, ‘target‘, and ‘interaction‘.
primary_thresh	Numeric. Absolute score cutoff for primary node selection.
secondary_thresh	Numeric. Absolute score cutoff for secondary node restriction.
upstream_input	A named numeric vector of upstream seed nodes.
RNA_input	Optional named numeric vector of differential expression values; merged into ATT.

Value

A list with: - SIF: data.frame of filtered edges (‘source‘,‘target‘,‘interaction‘). - ATT: data.frame of node attributes (‘source‘,‘score‘,‘level‘,‘type‘,‘RNA_input‘).

Examples

```
dec_res <- data.frame(
  source = paste0("G", 1:6),
  score  = c(2.5, 1.2, 0.8, -2.2, 1.5, -0.5),
  level  = c(0, 0, 1, 0, 1, 1)
)
meta_net <- data.frame(
  source      = c("G1","G1","G2","G3","G4","G5"),
  target      = c("G2","G3","G4","G5","G2","G6"),
  interaction = c(1, -1, 1, 1, -1, 1)
)
upstream_input <- c(G1 = 1, G4 = -1)
RNA_input <- c(G1 = 0.5, G2 = -0.2, G4 = 1.1)
dbl_net <- reduce_solution_network_double_thresh(
  decoupleRnival_res = dec_res,
  meta_network       = meta_net,
  primary_thresh     = 2,
  secondary_thresh   = 1,
  upstream_input     = upstream_input,
  RNA_input          = RNA_input
)
print(dbl_net$SIF)
print(dbl_net$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)
```

`translate_column_HMDB` *Translate Column Using HMDB Mapper*

Description

This function translates the values in a column using a provided Human Metabolome Database (HMDB) mapper vector. It modifies the input values by replacing certain prefixes and suffixes according to specific rules.

Usage

```
translate_column_HMDB(my_column, HMDB_mapper_vec)
```

Arguments

`my_column`	A vector of values to be translated.
`HMDB_mapper_vec`	A named vector where the names are the original identifiers and the values are the corresponding HMDB identifiers.

Value

A vector with the translated values.

Examples

```
# Create a sample column and HMDB mapper vector
my_column <- c("Metab__1234_a", "Gene5678_b", "Metab__91011_c")
HMDB_mapper_vec <- c("1234" = "HMDB00001", "5678" = "HMDB00002", "91011" = "HMDB00003")

# Translate the column
translated_column <- translate_column_HMDB(my_column, HMDB_mapper_vec)
```

translate_res *translate_res*

Description

formats the network with readable names

Usage

```
translate_res(SIF, ATT, HMDB_mapper_vec = NULL)
```

Arguments

SIF	result SIF of decoupleRnival pipeline
ATT	result ATT of decoupleRnival pipeline
HMDB_mapper_vec	a named vector with HMDB Ids as names and desired metabolite names as values.

Value

list with network and attribute tables.

Examples

```
# Create a meta network data frame
example_SIF <- data.frame(
source = c("GPX1", "Gene863__GPX1"),
target = c("Gene863__GPX1", "Metab__HMDB0003337_c"),
sign = c(1, 1)
)

example_ATT <- data.frame(
Nodes = c("GPX1", "Gene863__GPX1","Metab__HMDB0003337_c"),
sign = c(1, 1, 1)
)

example_SIF

data("HMDB_mapper_vec")

translated_res <- translate_res(example_SIF,example_ATT,HMDB_mapper_vec)

translated_res$SIF
```

`wide_ulm_res` *Convert ULM Results to Wide Format*

Description

This function converts the results from a ULM analysis to a wide format data frame. The input is a data frame with columns for source, condition, and score. The output is a data frame where each row represents a source and each column represents a condition, with the corresponding scores as values.

Usage

```
wide_ulm_res(ulm_result)
```

Arguments

`ulm_result`	A data frame representing the ULM results with columns: source, condition, and score.

Value

A data frame in wide format where each row is a source and each column is a condition.

Examples

```
# Create a sample ULM result
ulm_result <- data.frame(source = c("A", "A", "B", "B"),
                         condition = c("cond1", "cond2", "cond1", "cond2"),
                         score = c(0.5, 0.8, 0.3, 0.7))

# Convert to wide format
wide_ulm_result <- wide_ulm_res(ulm_result)
```

Index

∗ **datasets**
HMDB_mapper_vec, 16
meta_network, 18
toy_metabolic_input, 31
toy_network, 32
toy_RNA, 33
toy_signaling_input, 33

compress_same_children, 3
cosmos_data, *4*, 4, *26*, *29–31*
createLinearColors, 5

decompress_moon_result, 6
decompress_solution_network, 6
decoupleRnival, 8
default_CARNIVAL_options, 9, *22*, *25*, *29*, *30*
display_node_neighboorhood, 10

extract_nodes_for_ORA, 11

filter_incohrent_TF_target, 12
format_COSMOS_res, 13
format_LR_ressource, 14

get_moon_scoring_network, 15
gmt_to_dataframe, 16

HMDB_mapper_vec, 16

load_tf_regulon_dorothea, *4*, 17, *22–25*

make_heatmap_color_palette, 17
meta_network, *4*, 18, *22–25*
meta_network_cleanup, 19
moon, 19

prepare_metab_inputs, 21
preprocess_COSMOS_metabolism_to_signaling, *4*, 21, *26*, *29*, *31*
preprocess_COSMOS_signaling_to_metabolism, *4*, 23, *26*, *30*
print, *26*
print.cosmos_data, 25

reduce_solution_network, 26
reduce_solution_network_double_thresh, 27
run_COSMOS_metabolism_to_signaling, *9*, 29
run_COSMOS_signaling_to_metabolism, *9*, 30
runCARNIVAL, *10*, *23*, *25*, *29*, *31*

sign, *4*, *22*, *24*

toy_metabolic_input, 31
toy_network, 32
toy_RNA, 33
toy_signaling_input, 33
translate_column_HMDB, 34
translate_res, 35

wide_ulm_res, 36