Package 'decoupleR'

January 19, 2025

Type Package

Title decoupleR: Ensemble of computational methods to infer biological activities from omics data

Version 2.12.0

Description Many methods allow us to extract biological activities from omics data using information from prior knowledge resources, reducing the dimensionality for increased statistical power and better interpretability. Here, we present decoupleR, a Bioconductor package containing different statistical methods to extract these signatures within a unified framework. decoupleR allows the user to flexibly test any method with any resource. It incorporates methods that take into account the sign and weight of network interactions. decoupleR can be used with any omic, as long as its features can be linked to a biological process based on prior knowledge. For example, in transcriptomics gene sets regulated by a transcription factor, or in phospho-proteomics phosphosites that are targeted by a kinase.

License GPL-3 + file LICENSE

URL https://saezlab.github.io/decoupleR/

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

Depends R (>= 4.0)

Imports BiocParallel, broom, dplyr, magrittr, Matrix, parallelly, purrr, rlang, stats, stringr, tibble, tidyr, tidyselect, withr

Suggests glmnet (>= 4.1-7), GSVA, viper, fgsea (>= 1.15.4), AUCell, SummarizedExperiment, rpart, ranger, BiocStyle, covr, knitr, pkgdown, RefManageR, rmarkdown, roxygen2, sessioninfo, pheatmap, testthat, OmnipathR, Seurat, ggplot2, ggrepel, patchwork, magick

VignetteBuilder knitr

biocViews DifferentialExpression, FunctionalGenomics, GeneExpression, GeneRegulation, Network, Software, StatisticalMethod, Transcription,

Config/testthat/edition 3

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.0

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

git_branch RELEASE_3_20

git_last_commit bc3245c

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2025-01-19

Author Pau Badia-i-Mompel [aut, cre] (<https://orcid.org/0000-0002-1004-3923>),
Jesús Vélez-Santiago [aut] (<https://orcid.org/0000-0001-5128-3838>),
Jana Braunger [aut] (<https://orcid.org/0000-0003-0820-9987>),
Celina Geiss [aut] (<https://orcid.org/0000-0002-8740-706X>),
Daniel Dimitrov [aut] (<https://orcid.org/0000-0002-5197-2112>),
Sophia Müller-Dott [aut] (<https://orcid.org/0000-0002-9710-1865>),
Petr Taus [aut] (<https://orcid.org/0000-0003-3764-9033>),
Aurélien Dugourd [aut] (<https://orcid.org/0000-0002-0714-028X>),
Christian H. Holland [aut] (<https://orcid.org/0000-0002-3060-5786>),
Ricardo O. Ramirez Flores [aut]
(<https://orcid.org/0000-0003-0087-371X>),
Julio Saez-Rodriguez [aut] (<https://orcid.org/0000-0002-8552-8976>)

Maintainer Pau Badia-i-Mompel <pau.badia@uni-heidelberg.de>

Contents

decoupleR-package . . . 3
.decoupler_mat_format . . . 4
.decoupler_network_format . . . 4
.fit_preprocessing . . . 5
check_corr . . . 5
convert_f_defaults . . . 6
decouple . . . 7
extract_sets . . . 9
filt_minsize . . . 10
get_collectri . . . 10
get_dorothea . . . 11
get_ksn_omnipath . . . 12
get_profile_of . . . 12
get_progeny . . . 13
get_resource . . . 14
get_toy_data . . . 14
intersect_regulons . . . 15
pivot_wider_profile . . . 15
randomize_matrix . . . 17
rename_net . . . 18
run_aucell . . . 19
run_consensus . . . 20
run_fgsea . . . 21
run_gsva . . . 22
run_mdt . . . 25
run_mlm . . . 26

run_ora 28
run_udt 30
run_ulm 31
run_viper 33
run_wmean 34
run_wsum 36
show_methods 37
show_resources 38
tidyeval 38
%>% 39

Index **40**

`decoupleR-package` *decoupleR: decoupleR: Ensemble of computational methods to infer biological activities from omics data*

Description

Many methods allow us to extract biological activities from omics data using information from prior knowledge resources, reducing the dimensionality for increased statistical power and better interpretability. Here, we present decoupleR, a Bioconductor package containing different statistical methods to extract these signatures within a unified framework. decoupleR allows the user to flexibly test any method with any resource. It incorporates methods that take into account the sign and weight of network interactions. decoupleR can be used with any omic, as long as its features can be linked to a biological process based on prior knowledge. For example, in transcriptomics gene sets regulated by a transcription factor, or in phospho-proteomics phosphosites that are targeted by a kinase.

Author(s)

Maintainer: Pau Badia-i-Mompel <pau.badia@uni-heidelberg.de> (ORCID)

Authors:

- Jesús Vélez-Santiago <jvelezmagic@gmail.com> (ORCID)
- Jana Braunger <jana.bc@gmx.de> (ORCID)
- Celina Geiss <celina.geiss@stud.uni-heidelberg.de> (ORCID)
- Daniel Dimitrov <daniel.dimitrov@uni-heidelberg.de> (ORCID)
- Sophia Müller-Dott <sophia.mueller-dott@uni-heidelberg.de> (ORCID)
- Petr Taus <petr.taus@ceitec.muni.cz> (ORCID)
- Aurélien Dugourd <aurelien.dugourd@bioquant.uni-heidelberg.de> (ORCID)
- Christian H. Holland <cholland2408@gmail.com> (ORCID)
- Ricardo O. Ramirez Flores <roramirezf@uni-heidelberg.de > (ORCID)
- Julio Saez-Rodriguez <pub.saez@uni-heidelberg.de> (ORCID)

See Also

Useful links:

- https://saezlab.github.io/decoupleR/
- Report bugs at https://github.com/saezlab/decoupleR/issues

.decoupler_mat_format *DecoupleR mat format*

Description

DecoupleR mat format

Arguments

mat	Matrix to evaluate (e.g. expression matrix). Target nodes in rows and conditions in columns. rownames(mat) must have at least one intersection with the elements in network .target column.

See Also

Other decoupleR formats: .decoupler_network_format

.decoupler_network_format *DecoupleR network format*

Description

A network passed to any run_ method in the package must contain at least two attributes: .source and .target. In addition, the methods must map their corresponding metadata associated with their edges.

Arguments

network	Tibble or dataframe with edges and it's associated metadata.
.source	Column with source nodes.
.target	Column with target nodes.
.mor	Column with edge mode of regulation (i.e. mor).
.likelihood	Deprecated argument. Now it will always be set to 1.

Details

- All the attributes to be mapped are prefixed by .
- The idea of using this type of mapping is to provide flexibility to different types of networks, be they regulatory, metabolic, or of any other type. This way, you should only consider having your network or networks in a long format and these can easily be manipulated by functions within the tidyverse ecosystem.

See Also

Other decoupleR formats: .decoupler_mat_format

.fit_preprocessing *Pre-processing for methods that fit networks.*

Description

- If `center` is true, then the expression values are centered by the mean of expression across the samples.

Usage

```
.fit_preprocessing(network, mat, center, na.rm, sparse)
```

Arguments

`network`	Tibble or dataframe with edges and it's associated metadata.
`mat`	Matrix to evaluate (e.g. expression matrix). Target nodes in rows and conditions in columns. `rownames(mat)` must have at least one intersection with the elements in `network .target` column.
`center`	Logical value indicating if `mat` must be centered by `base::rowMeans()`.
`na.rm`	Should missing values (including NaN) be omitted from the calculations of `base::rowMeans()`?
`sparse`	Deprecated parameter.

Value

A named list of matrices to evaluate in methods that fit models, like `.mlm_analysis()`.

- mat: Features as rows and samples as columns.
- mor_mat: Features as rows and columns as source.

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")
mat <- readRDS(file.path(inputs_dir, "mat.rds"))
net <- readRDS(file.path(inputs_dir, "net.rds"))
net <- rename_net(net, source, target, mor)
.fit_preprocessing(net, mat, center = FALSE, na.rm = FALSE, sparse = FALSE)
```

check_corr *Check correlation (colinearity)*

Description

Checks the correlation across the regulators in a network.

Usage

```
check_corr(
  network,
  .source = "source",
  .target = "target",
  .mor = "mor",
  .likelihood = NULL
)
```

Arguments

network	Tibble or dataframe with edges and it's associated metadata.
.source	Column with source nodes.
.target	Column with target nodes.
.mor	Column with edge mode of regulation (i.e. mor).
.likelihood	Deprecated argument. Now it will always be set to 1.

Value

Correlation pairs tibble.

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")
net <- readRDS(file.path(inputs_dir, "net.rds"))
check_corr(net, .source='source')
```

---

convert_f_defaults *Rename columns and add defaults values if column not present*

Description

convert_f_defaults() combine the dplyr::rename() way of working and with the tibble::add_column() to add columns with default values in case they don't exist after renaming data.

Usage

```
convert_f_defaults(.data, ..., .def_col_val = c(), .use_dots = TRUE)
```

Arguments

.data	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
...	For rename(): <tidy-select> Use new_name = old_name to rename selected variables. For rename_with(): additional arguments passed onto .fn.
.def_col_val	Named vector with columns with default values if none exist after rename.
.use_dots	Should a dot prefix be added to renamed variables? This will allow swapping of columns.

Details

The objective of using .use_dots is to be able to swap columns which, by default, is not allowed by the `dplyr::rename()` function. The same behavior can be replicated by simply using the `dplyr::select()`, however, the select evaluation allows much more flexibility so that unexpected results could be obtained. Despite this, a future implementation will consider this form of execution to allow renaming the same column to multiple ones (i.e. extend dataframe extension).

Value

An object of the same type as .data. The output has the following properties:

- Rows are not affected.
- Column names are changed.
- Column order is the same as that of the function call.

Examples

```
df <- tibble::tibble(x = 1, y = 2, z = 3)

# Rename columns
df <- tibble::tibble(x = 1, y = 2)
convert_f_defaults(
    .data = df,
    new_x = x,
    new_y = y,
    new_z = NULL,
    .def_col_val = c(new_z = 3)
)
```

---

decouple *Evaluate multiple statistics with same input data*

Description

Calculate the source activity per sample out of a gene expression matrix by coupling a regulatory network with a variety of statistics.

Usage

```
decouple(
  mat,
  network,
  .source = source,
  .target = target,
  statistics = NULL,
  args = list(NULL),
  consensus_score = TRUE,
  consensus_stats = NULL,
  include_time = FALSE,
  show_toy_call = FALSE,
  minsize = 5
)
```

Arguments

mat	Matrix to evaluate (e.g. expression matrix). Target nodes in rows and conditions in columns. rownames(mat) must have at least one intersection with the elements in network .target column.
network	Tibble or dataframe with edges and it's associated metadata.
.source	Column with source nodes.
.target	Column with target nodes.
statistics	Statistical methods to be run sequentially. If none are provided, only top performer methods are run (mlm, ulm and wsum).
args	A list of argument-lists the same length as statistics (or length 1). The default argument, list(NULL), will be recycled to the same length as statistics, and will call each function with no arguments (apart from mat, network, .source and, .target).
consensus_score	Boolean whether to run a consensus score between methods.
consensus_stats	List of estimate names to use for the calculation of the consensus score. This is used to filter out extra estimations from some methods, for example wsum returns wsum, corr_wsum and norm_wsum. If none are provided, and also no statstics where provided, only top performer methods are used (mlm, ulm and norm_wsum). Else, it will use all available estimates after running all methods in the statistics argument.
include_time	Should the time per statistic evaluated be informed?
show_toy_call	The call of each statistic must be informed?
minsize	Integer indicating the minimum number of targets per source.

Value

A long format tibble of the enrichment scores for each source across the samples. Resulting tibble contains the following columns:

1. run_id: Indicates the order in which the methods have been executed.
2. statistic: Indicates which method is associated with which score.
3. source: Source nodes of network.
4. condition: Condition representing each column of mat.
5. score: Regulatory activity (enrichment score).
6. statistic_time: If requested, internal execution time indicator.
7. p_value: p-value (if available) of the obtained score.

See Also

Other decoupleR statistics: run_aucell(), run_fgsea(), run_gsva(), run_mdt(), run_mlm(), run_ora(), run_udt(), run_ulm(), run_viper(), run_wmean(), run_wsum()

Examples

```
if (FALSE) {
    inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")

    mat <- readRDS(file.path(inputs_dir, "mat.rds"))
    net <- readRDS(file.path(inputs_dir, "net.rds"))

    decouple(
        mat = mat,
        network = net,
        .source = "source",
        .target = "target",
        statistics = c("gsva", "wmean", "wsum", "ulm", "aucell"),
        args = list(
            gsva = list(verbose = FALSE),
            wmean = list(.mor = "mor", .likelihood = "likelihood"),
            wsum = list(.mor = "mor"),
            ulm = list(.mor = "mor")
        ),
        minsize = 0
    )
}
```

`extract_sets` *Extract sets*

Description

Extracts feature sets from a renamed network (see rename_net).

Usage

```
extract_sets(network)
```

Arguments

`network`	Tibble or dataframe with edges and it's associated metadata.

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")
mat <- readRDS(file.path(inputs_dir, "mat.rds"))
net <- readRDS(file.path(inputs_dir, "net.rds"))
net <- rename_net(net, source, target, mor)
extract_sets(net)
```

`filt_minsize` *Filter sources with minsize targets*

Description

Filter sources of a net with less than minsize targets

Usage

```
filt_minsize(mat_f_names, network, minsize = 5)
```

Arguments

`mat_f_names`	Feature names of mat.
`network`	Tibble or dataframe with edges and it's associated metadata.
`minsize`	Integer indicating the minimum number of targets per source.

Value

Filtered network.

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")
mat <- readRDS(file.path(inputs_dir, "mat.rds"))
net <- readRDS(file.path(inputs_dir, "net.rds"))
net <- rename_net(net, source, target, mor)
filt_minsize(rownames(mat), net, minsize = 4)
```

`get_collectri` *CollecTRI gene regulatory network. Wrapper to access CollecTRI gene regulatory network. CollecTRI is a comprehensive resource containing a curated collection of transcription factors (TFs) and their target genes. It is an expansion of DoRothEA. Each interaction is weighted by its mode of regulation (either positive or negative).*

Description

CollecTRI gene regulatory network. Wrapper to access CollecTRI gene regulatory network. CollecTRI is a comprehensive resource containing a curated collection of transcription factors (TFs) and their target genes. It is an expansion of DoRothEA. Each interaction is weighted by its mode of regulation (either positive or negative).

Usage

```
get_collectri(organism = "human", split_complexes = FALSE, ...)
```

Arguments

organism	Which organism to use. Only human, mouse and rat are available.
split_complexes	Whether to split complexes into subunits. By default complexes are kept as they are.
...	Ignored.

Examples

```
collectri <- get_collectri(organism='human', split_complexes=FALSE)
```

---

get_dorothea *DoRothEA gene regulatory network.*

Description

Wrapper to access DoRothEA gene regulatory network. DoRothEA is a comprehensive resource containing a curated collection of transcription factors (TFs) and their target genes. Each interaction is weighted by its mode of regulation (either positive or negative) and by its confidence level

Usage

```
get_dorothea(
  organism = "human",
  levels = c("A", "B", "C"),
  weight_dict = list(A = 1, B = 2, C = 3, D = 4)
)
```

Arguments

organism	Which organism to use. Only human, mouse and rat are available.
levels	List of confidence levels to return. Goes from A to D, A being the most confident and D being the less.
weight_dict	Dictionary of values to divide the mode of regulation (-1 or 1), one for each confidence level. Bigger values will generate weights close to zero.

Examples

```
dorothea <- get_dorothea(organism='human', levels=c('A', 'B'))
```

get_ksn_omnipath *OmniPath kinase-substrate network*

Description

Retrieve a ready to use, curated kinase-substrate Network from the OmniPath database.

Usage

```
get_ksn_omnipath(...)
```

Arguments

...	Passed to OmnipathR::import_omnipath_enzsub.

Details

Import enzyme-PTM network from OmniPath, then filter out anything that is not phospho or dephosphorilation. Then format the columns for use with decoupleR functions.

get_profile_of *Complete a data frame with missing combinations of data*

Description

Turns implicit missing values into explicit missing values. This is a wrapper around expand(), dplyr::full_join() and replace_na() that's useful for completing missing combinations of data.

Usage

```
get_profile_of(data, sources, values_fill = NA)
```

Arguments

data	A data frame.
sources	A named vector or list with the values to expand and get profile.
values_fill	Optionally, a (scalar) value that specifies what each value should be filled in with when missing. This can be a named list if you want to apply different fill values to different value columns.

Value

A data frame with the expanded grid of the values passed in sources and filled as specified in the fill argument.

See Also

complete expand

Examples

```
## Not run:
library(dplyr, warn.conflicts = FALSE)
df <- tibble(
    group = c(1:2, 1),
    item_id = c(1:2, 2),
    item_name = c("a", "b", "b"),
    value1 = 1:3,
    value2 = 4:6
)

to_get_profile <- list(group = c(1, 2, 3), item_id = c(1, 2))

# This will add the combinations of group 3 with the id of the items
df %>% get_profile_of(sources = to_get_profile)

# You can also choose to fill in missing values

# This only fill with "Unknown" the NA values of the column item_name
df %>% get_profile_of(
    sources = to_get_profile,
    values_fill = list(item_name = "Unknown")
)

# Replace all NAs with "Unkwnon"
df %>% get_profile_of(sources = to_get_profile, values_fill = "Unknown")

## End(Not run)
```

`get_progeny` *Pathway RespOnsive GENes for activity inference (PROGENy).*

Description

Wrapper to access PROGENy model gene weights. Each pathway is defined with a collection of target genes, each interaction has an associated p-value and weight. The top significant interactions per pathway are returned.

Usage

```
get_progeny(organism = "human", top = 500)
```

Arguments

`organism`	Which organism to use. Only human and mouse are available.
`top`	Number of genes per pathway to return.

Examples

```
progeny <- get_progeny(organism='human', top=500)
```

`get_resource` *Wrapper to access resources inside Omnipath. This wrapper allows to easily query different prior knowledge resources. To check available resources run* `decoupleR::show_resources()`. *For more information visit the official website for* R*hrefhttps://omnipathdb.org/Omnipath.*

Description

Wrapper to access resources inside Omnipath. This wrapper allows to easily query different prior knowledge resources. To check available resources run `decoupleR::show_resources()`. For more information visit the official website for Omnipath.

Usage

```
get_resource(name, organism = "human", ...)
```

Arguments

`name`	Name of the resource to query.
`organism`	Organism name or NCBI Taxonomy ID.
`...`	Passed to `OmnipathR::import_omnipath_annotations`.

Examples

```
df <- decoupleR::get_resource('SIGNOR')
```

`get_toy_data` *Generate a toy* `mat` *and* `network`.

Description

Generate a toy `mat` and `network`.

Usage

```
get_toy_data(n_samples = 24, seed = 42)
```

Arguments

`n_samples`	Number of samples to simulate.
`seed`	A single value, interpreted as an integer, or NULL for random number generation.

Value

List containing `mat` and `network`.

Examples

```
data <- get_toy_data()
mat <- data$mat
network <- data$network
```

`intersect_regulons` *Intersect network target features with input matrix.*

Description

Keep only edges which its target features belong to the input matrix.

Usage

```
intersect_regulons(mat, network, .source, .target, minsize)
```

Arguments

`mat`	Matrix to evaluate (e.g. expression matrix). Target nodes in rows and conditions in columns. `rownames(mat)` must have at least one intersection with the elements in `network` `.target` column.
`network`	Tibble or dataframe with edges and it's associated metadata.
`.source`	Column with source nodes.
`.target`	Column with target nodes.
`minsize`	Minimum number of targets per source allowed.

Value

Filtered tibble.

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")
mat <- readRDS(file.path(inputs_dir, "mat.rds"))
net <- readRDS(file.path(inputs_dir, "net.rds"))
intersect_regulons(mat, net, source, target, minsize=4)
```

`pivot_wider_profile` *Pivot a data frame to wider and convert it to matrix*

Description

Generates a kind of table where the rows come from `id_cols`, the columns from `names_from` and the values from `values_from`.

Usage

```
pivot_wider_profile(
  data,
  id_cols,
  names_from,
  values_from,
  values_fill = NA,
  to_matrix = FALSE,
  to_sparse = FALSE,
  ...
)
```

Arguments

data	A data frame to pivot.
id_cols	<tidy-select> A set of columns that uniquely identify each observation. Typically used when you have redundant variables, i.e. variables whose values are perfectly correlated with existing variables. Defaults to all columns in data except for the columns specified through names_from and values_from. If a tidyselect expression is supplied, it will be evaluated on data after removing the columns specified through names_from and values_from.
names_from, values_from	<tidy-select> A pair of arguments describing which column (or columns) to get the name of the output column (names_from), and which column (or columns) to get the cell values from (values_from). If values_from contains multiple values, the value will be added to the front of the output column.
values_fill	Optionally, a (scalar) value that specifies what each value should be filled in with when missing. This can be a named list if you want to apply different fill values to different value columns.
to_matrix	Logical value indicating if the result should be a matrix. Parameter is ignored in case sparse is TRUE.
to_sparse	Logical value indicating whether the resulting matrix should be sparse or not.
...	Additional arguments passed on to methods.

Details

In the current state of the function, to ensure its operation, the id_cols parameter is a single selector.

Value

"widened" data; it is increasing the number of columns and decreasing the number of rows.

Examples

```
## Not run:
df <- tibble::tibble(
    tf = c("tf_1", "tf_1", "tf_2", "tf_2"),
    gene = c("gene_1", "gene_2", "gene_1", "gene_2"),
    mor = c(1, -1, 1, -1)
```

```
)

# Return a tibble
pivot_wider_profile(
    data = df,
    id_cols = tf,
    names_from = gene,
    values_from = mor
)

# Return a matrix
pivot_wider_profile(
    data = df,
    id_cols = tf,
    names_from = gene,
    values_from = mor,
    to_matrix = TRUE
)
# Return a sparse Matrix of class "dgCMatrix"
pivot_wider_profile(
    data = df,
    id_cols = tf,
    names_from = gene,
    values_from = mor,
    to_sparse = TRUE
)

## End(Not run)
```

`randomize_matrix` *Randomize matrix*

Description

Utility function used in functions that require permutations of the expression matrix

Usage

```
randomize_matrix(mat, randomize_type = c("rows", "cols_independently"))
```

Arguments

`mat`	Matrix to randomize.
`randomize_type`	How to randomize.

Value

Randomized matrix

Examples

```
## Not run:
mat <- matrix(seq_len(9), ncol = 3)
mat

set.seed(42)
randomize_matrix(mat, randomize_type = "rows")

set.seed(42)
randomize_matrix(mat, randomize_type = "cols_independently")

## End(Not run)
```

`rename_net` *Rename network*

Description

Renames a given network to these column names: .source, .target, .mor, If .mor is not provided, then the function sets them to default values.

Usage

```
rename_net(
  network,
  .source,
  .target,
  .mor = NULL,
  .likelihood = NULL,
  def_mor = 1
)
```

Arguments

`network`	Tibble or dataframe with edges and it's associated metadata.
`.source`	Column with source nodes.
`.target`	Column with target nodes.
`.mor`	Column with edge mode of regulation (i.e. mor).
`.likelihood`	Deprecated argument. Now it will always be set to 1.
`def_mor`	Default value for .mor when not provided.

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")
mat <- readRDS(file.path(inputs_dir, "mat.rds"))
net <- readRDS(file.path(inputs_dir, "net.rds"))
rename_net(net, source, target, mor)
```

`run_aucell` *AUCell*

Description

Calculates regulatory activities using AUCell.

Usage

```
run_aucell(
  mat,
  network,
  .source = source,
  .target = target,
  aucMaxRank = ceiling(0.05 * nrow(rankings)),
  nproc = availableCores(),
  seed = 42,
  minsize = 5
)
```

Arguments

`mat`	Matrix to evaluate (e.g. expression matrix). Target nodes in rows and conditions in columns. `rownames(mat)` must have at least one intersection with the elements in `network` `.target` column.
`network`	Tibble or dataframe with edges and it's associated metadata.
`.source`	Column with source nodes.
`.target`	Column with target nodes.
`aucMaxRank`	Threshold to calculate the AUC.
`nproc`	Number of cores to use for computation.
`seed`	A single value, interpreted as an integer, or NULL for random number generation.
`minsize`	Integer indicating the minimum number of targets per source.

Details

AUCell (Aibar et al., 2017) uses the Area Under the Curve (AUC) to calculate whether a set of targets is enriched within the molecular readouts of each sample. To do so, AUCell first ranks the molecular features of each sample from highest to lowest value, resolving ties randomly. Then, an AUC can be calculated using by default the top 5% molecular features in the ranking. Therefore, this metric, `aucell`, represents the proportion of abundant molecular features in the target set, and their relative abundance value compared to the other features within the sample.

Aibar S. et al. (2017) Scenic: single-cell regulatory network inference and clustering. Nat. Methods, 14, 1083–1086.

See Also

Other decoupleR statistics: `decouple()`, `run_fgsea()`, `run_gsva()`, `run_mdt()`, `run_mlm()`, `run_ora()`, `run_udt()`, `run_ulm()`, `run_viper()`, `run_wmean()`, `run_wsum()`

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")

mat <- readRDS(file.path(inputs_dir, "mat.rds"))
net <- readRDS(file.path(inputs_dir, "net.rds"))

run_aucell(mat, net, minsize=0, nproc=1, aucMaxRank=3)
```

---

run_consensus *Consensus score between methods*

Description

Function to generate a consensus score between methods from the result of the `decouple` function.

Usage

```
run_consensus(df, include_time = FALSE, seed = NULL)
```

Arguments

`df`	`decouple` data frame result
`include_time`	Should the time per statistic evaluated be informed?
`seed`	Deprecated parameter.

Value

Updated tibble with the computed consensus score between methods

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")
mat <- readRDS(file.path(inputs_dir, "mat.rds"))
net <- readRDS(file.path(inputs_dir, "net.rds"))

results <- decouple(
   mat = mat,
   network = net,
   .source = "source",
   .target = "target",
   statistics = c("wmean", "ulm"),
   args = list(
            wmean = list(.mor = "mor", .likelihood = "likelihood"),
            ulm = list(.mor = "mor", .likelihood = "likelihood")
        ),
   consensus_score = FALSE,
   minsize = 0
   )
run_consensus(results)
```

run_fgsea *Fast Gene Set Enrichment Analysis (FGSEA)*

Description

Calculates regulatory activities using FGSEA.

Usage

```
run_fgsea(
  mat,
  network,
  .source = source,
  .target = target,
  times = 100,
  nproc = availableCores(),
  seed = 42,
  minsize = 5,
  ...
)
```

Arguments

mat	Matrix to evaluate (e.g. expression matrix). Target nodes in rows and conditions in columns. `rownames(mat)` must have at least one intersection with the elements in `network .target` column.
network	Tibble or dataframe with edges and it's associated metadata.
.source	Column with source nodes.
.target	Column with target nodes.
times	How many permutations to do?
nproc	Number of cores to use for computation.
seed	A single value, interpreted as an integer, or NULL.
minsize	Integer indicating the minimum number of targets per source.
...	Arguments passed on to `fgsea::fgseaMultilevel` `sampleSize` The size of a random set of genes which in turn has size = pathwaySize `minSize` Minimal size of a gene set to test. All pathways below the threshold are excluded. `maxSize` Maximal size of a gene set to test. All pathways above the threshold are excluded. `eps` This parameter sets the boundary for calculating the p value. `scoreType` This parameter defines the GSEA score type. Possible options are ("std", "pos", "neg"). By default ("std") the enrichment score is computed as in the original GSEA. The "pos" and "neg" score types are intended to be used for one-tailed tests (i.e. when one is interested only in positive ("pos") or negateive ("neg") enrichment). `gseaParam` GSEA parameter value, all gene-level statis are raised to the power of 'gseaParam' before calculation of GSEA enrichment scores.

BPPARAM Parallelization parameter used in bplapply. Can be used to specify cluster to run. If not initialized explicitly or by setting ‘nproc‘ default value ‘bpparam()‘ is used.

absEps deprecated, use ‘eps‘ parameter instead

Details

GSEA (Aravind et al., 2005) starts by transforming the input molecular readouts in mat to ranks for each sample. Then, an enrichment score `fgsea` is calculated by walking down the list of features, increasing a running-sum statistic when a feature in the target feature set is encountered and decreasing it when it is not. The final score is the maximum deviation from zero encountered in the random walk. Finally, a normalized score `norm_fgsea`, can be obtained by computing the z-score of the estimate compared to a null distribution obtained from N random permutations. The used implementation is taken from the package `fgsea` (Korotkevich et al., 2021).

Aravind S. et al. (2005) Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. PNAS. 102, 43.

Korotkevich G. et al. (2021) Fast gene set enrichment analysis. bioRxiv. DOI: https://doi.org/10.1101/060012.

Value

A long format tibble of the enrichment scores for each source across the samples. Resulting tibble contains the following columns:

1. `statistic`: Indicates which method is associated with which score.
2. `source`: Source nodes of `network`.
3. `condition`: Condition representing each column of `mat`.
4. `score`: Regulatory activity (enrichment score).

See Also

Other decoupleR statistics: `decouple()`, `run_aucell()`, `run_gsva()`, `run_mdt()`, `run_mlm()`, `run_ora()`, `run_udt()`, `run_ulm()`, `run_viper()`, `run_wmean()`, `run_wsum()`

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")

mat <- readRDS(file.path(inputs_dir, "mat.rds"))
net <- readRDS(file.path(inputs_dir, "net.rds"))

run_fgsea(mat, net, minsize=0, nproc=1)
```

---

run_gsva *Gene Set Variation Analysis (GSVA)*

Description

Calculates regulatory activities using GSVA.

Usage

```
run_gsva(
  mat,
  network,
  .source = source,
  .target = target,
  verbose = FALSE,
  method = c("gsva", "plage", "ssgsea", "zscore"),
  minsize = 5L,
  maxsize = Inf,
  ...
)
```

Arguments

mat	Matrix to evaluate (e.g. expression matrix). Target nodes in rows and conditions in columns. `rownames(mat)` must have at least one intersection with the elements in `network .target` column.
network	Tibble or dataframe with edges and it's associated metadata.
.source	Column with source nodes.
.target	Column with target nodes.
verbose	Gives information about each calculation step. Default: FALSE.
method	Method to employ in the estimation of gene-set enrichment. scores per sample. By default this is set to gsva (Hänzelmann et al, 2013). Further available methods are "plage", "ssgsea" and "zscore". Read more in the manual of `GSVA::gsva`.
minsize	Integer indicating the minimum number of targets per source. Must be greater than 0.
maxsize	Integer indicating the maximum number of targets per source.
...	Arguments passed on to `GSVA::gsvaParam`, `GSVA::ssgseaParam` `assay` The name of the assay to use in case `exprData` is a multi-assay container, otherwise ignored. By default, the first assay is used. `annotation` The name of a Bioconductor annotation package for the gene identifiers occurring in the row names of the expression data matrix. This can be used to map gene identifiers occurring in the gene sets if those are provided in a `GeneSetCollection`. By default gene identifiers used in expression data matrix and gene sets are matched directly. `kcdf` Character vector of length 1 denoting the kernel to use during the non-parametric estimation of the cumulative distribution function of expression levels across samples. By default, `kcdf="Gaussian"` which is suitable when input expression values are continuous, such as microarray fluorescent units in logarithmic scale, RNA-seq log-CPMs, log-RPKMs or log-TPMs. When input expression values are integer counts, such as those derived from RNA-seq experiments, then this argument should be set to `kcdf="Poisson"`. `tau` Numeric vector of length 1. The exponent defining the weight of the tail in the random walk performed by the `GSVA` (Hänzelmann et al., 2013) method. The default value is 1 as described in the paper.

`maxDiff` Logical vector of length 1 which offers two approaches to calculate the enrichment statistic (ES) from the KS random walk statistic.

- `FALSE`: ES is calculated as the maximum distance of the random walk from 0.
- `TRUE` (the default): ES is calculated as the magnitude difference between the largest positive and negative random walk deviations.

`absRanking` Logical vector of length 1 used only when `maxDiff=TRUE`. When `absRanking=FALSE` (default) a modified Kuiper statistic is used to calculate enrichment scores, taking the magnitude difference between the largest positive and negative random walk deviations. When `absRanking=TRUE` the original Kuiper statistic that sums the largest positive and negative random walk deviations, is used. In this latter case, gene sets with genes enriched on either extreme (high or low) will be regarded as 'highly' activated.

`alpha` Numeric vector of length 1. The exponent defining the weight of the tail in the random walk performed by the `ssGSEA` (Barbie et al., 2009) method. The default value is 0.25 as described in the paper.

`normalize` Logical vector of length 1; if `TRUE` runs the `ssGSEA` method from Barbie et al. (2009) normalizing the scores by the absolute difference between the minimum and the maximum, as described in their paper. Otherwise this last normalization step is skipped.

Details

GSVA (Hänzelmann et al., 2013) starts by transforming the input molecular readouts in mat to a readout-level statistic using Gaussian kernel estimation of the cumulative density function. Then, readout-level statistics are ranked per sample and normalized to up-weight the two tails of the rank distribution. Afterwards, an enrichment score `gsva` is calculated using a running sum statistic that is normalized by subtracting the largest negative estimate from the largest positive one.

Hänzelmann S. et al. (2013) GSVA: gene set variation analysis for microarray and RNA-seq data. BMC Bioinformatics, 14, 7.

Value

A long format tibble of the enrichment scores for each source across the samples. Resulting tibble contains the following columns:

1. `statistic`: Indicates which method is associated with which score.
2. `source`: Source nodes of `network`.
3. `condition`: Condition representing each column of `mat`.
4. `score`: Regulatory activity (enrichment score).

See Also

Other decoupleR statistics: `decouple()`, `run_aucell()`, `run_fgsea()`, `run_mdt()`, `run_mlm()`, `run_ora()`, `run_udt()`, `run_ulm()`, `run_viper()`, `run_wmean()`, `run_wsum()`

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")

mat <- readRDS(file.path(inputs_dir, "mat.rds"))
net <- readRDS(file.path(inputs_dir, "net.rds"))
```

```
run_gsva(mat, net, minsize=1, verbose = FALSE)
```

`run_mdt` *Multivariate Decision Trees (MDT)*

Description

Calculates regulatory activities using MDT.

Usage

```
run_mdt(
  mat,
  network,
  .source = source,
  .target = target,
  .mor = mor,
  .likelihood = likelihood,
  sparse = FALSE,
  center = FALSE,
  na.rm = FALSE,
  trees = 10,
  min_n = 20,
  nproc = availableCores(),
  seed = 42,
  minsize = 5
)
```

Arguments

`mat`	Matrix to evaluate (e.g. expression matrix). Target nodes in rows and conditions in columns. `rownames(mat)` must have at least one intersection with the elements in `network` `.target` column.
`network`	Tibble or dataframe with edges and it's associated metadata.
`.source`	Column with source nodes.
`.target`	Column with target nodes.
`.mor`	Column with edge mode of regulation (i.e. mor).
`.likelihood`	Deprecated argument. Now it will always be set to 1.
`sparse`	Deprecated parameter.
`center`	Logical value indicating if `mat` must be centered by `base::rowMeans()`.
`na.rm`	Should missing values (including NaN) be omitted from the calculations of `base::rowMeans()`?
`trees`	An integer for the number of trees contained in the ensemble.
`min_n`	An integer for the minimum number of data points in a node that are required for the node to be split further.
`nproc`	Number of cores to use for computation.
`seed`	A single value, interpreted as an integer, or NULL for random number generation.
`minsize`	Integer indicating the minimum number of targets per source.

Details

MDT fits a multivariate regression random forest for each sample, where the observed molecular readouts in mat are the response variable and the regulator weights in net are the covariates. Target features with no associated weight are set to zero. The obtained feature importances from the fitted model are the activities `mdt` of the regulators in net.

Value

A long format tibble of the enrichment scores for each source across the samples. Resulting tibble contains the following columns:

1. `statistic`: Indicates which method is associated with which score.
2. `source`: Source nodes of `network`.
3. `condition`: Condition representing each column of `mat`.
4. `score`: Regulatory activity (enrichment score).

See Also

Other decoupleR statistics: `decouple()`, `run_aucell()`, `run_fgsea()`, `run_gsva()`, `run_mlm()`, `run_ora()`, `run_udt()`, `run_ulm()`, `run_viper()`, `run_wmean()`, `run_wsum()`

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")

mat <- readRDS(file.path(inputs_dir, "mat.rds"))
net <- readRDS(file.path(inputs_dir, "net.rds"))

run_mdt(mat, net, minsize=0)
```

---

run_mlm *Multivariate Linear Model (MLM)*

Description

Calculates regulatory activities using MLM.

Usage

```
run_mlm(
  mat,
  network,
  .source = source,
  .target = target,
  .mor = mor,
  .likelihood = likelihood,
  sparse = FALSE,
  center = FALSE,
  na.rm = FALSE,
  minsize = 5
)
```

Arguments

mat	Matrix to evaluate (e.g. expression matrix). Target nodes in rows and conditions in columns. rownames(mat) must have at least one intersection with the elements in network .target column.
network	Tibble or dataframe with edges and it's associated metadata.
.source	Column with source nodes.
.target	Column with target nodes.
.mor	Column with edge mode of regulation (i.e. mor).
.likelihood	Deprecated argument. Now it will always be set to 1.
sparse	Deprecated parameter.
center	Logical value indicating if mat must be centered by base::rowMeans().
na.rm	Should missing values (including NaN) be omitted from the calculations of base::rowMeans()?
minsize	Integer indicating the minimum number of targets per source.

Details

MLM fits a multivariate linear model for each sample, where the observed molecular readouts in mat are the response variable and the regulator weights in net are the covariates. Target features with no associated weight are set to zero. The obtained t-values from the fitted model are the activities (mlm) of the regulators in net.

Value

A long format tibble of the enrichment scores for each source across the samples. Resulting tibble contains the following columns:

1. statistic: Indicates which method is associated with which score.
2. source: Source nodes of network.
3. condition: Condition representing each column of mat.
4. score: Regulatory activity (enrichment score).

See Also

Other decoupleR statistics: decouple(), run_aucell(), run_fgsea(), run_gsva(), run_mdt(), run_ora(), run_udt(), run_ulm(), run_viper(), run_wmean(), run_wsum()

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")

mat <- readRDS(file.path(inputs_dir, "mat.rds"))
net <- readRDS(file.path(inputs_dir, "net.rds"))

run_mlm(mat, net, minsize=0)
```

run_ora *Over Representation Analysis (ORA)*

Description

Calculates regulatory activities using ORA.

Usage

```
run_ora(
  mat,
  network,
  .source = source,
  .target = target,
  n_up = ceiling(0.05 * nrow(mat)),
  n_bottom = 0,
  n_background = 20000,
  with_ties = TRUE,
  seed = 42,
  minsize = 5,
  ...
)
```

Arguments

mat	Matrix to evaluate (e.g. expression matrix). Target nodes in rows and conditions in columns. `rownames(mat)` must have at least one intersection with the elements in `network .target` column.
network	Tibble or dataframe with edges and it's associated metadata.
.source	Column with source nodes.
.target	Column with target nodes.
n_up	Integer indicating the number of top targets to slice from mat.
n_bottom	Integer indicating the number of bottom targets to slice from mat.
n_background	Integer indicating the background size of the sliced targets. If not specified the number of background targets is determined by the total number of unique targets in the union of `mat` and `network`.
with_ties	Should ties be kept together? The default, `TRUE`, may return more rows than you request. Use `FALSE` to ignore ties, and return the first `n` rows.
seed	A single value, interpreted as an integer, or NULL for random number generation.
minsize	Integer indicating the minimum number of targets per source.
...	Arguments passed on to `stats::fisher.test` `workspace` an integer specifying the size of the workspace used in the network algorithm. In units of 4 bytes. Only used for non-simulated p-values larger than 2×2 tables. Since R version 3.5.0, this also increases the internal stack size which allows larger problems to be solved, however sometimes needing hours. In such cases, `simulate.p.values=TRUE` may be more reasonable.

`hybrid` a logical. Only used for larger than 2×2 tables, in which cases it indicates whether the exact probabilities (default) or a hybrid approximation thereof should be computed.

`hybridPars` a numeric vector of length 3, by default describing "Cochran's conditions" for the validity of the chisquare approximation, see 'Details'.

`control` a list with named components for low level algorithm control. At present the only one used is `"mult"`, a positive integer ≥ 2 with default 30 used only for larger than 2×2 tables. This says how many times as much space should be allocated to paths as to keys: see file '`fexact.c`' in the sources of this package.

`or` the hypothesized odds ratio. Only used in the 2×2 case.

`alternative` indicates the alternative hypothesis and must be one of `"two.sided"`, `"greater"` or `"less"`. You can specify just the initial letter. Only used in the 2×2 case.

`conf.int` logical indicating if a confidence interval for the odds ratio in a 2×2 table should be computed (and returned).

`conf.level` confidence level for the returned confidence interval. Only used in the 2×2 case and if `conf.int = TRUE`.

`simulate.p.value` a logical indicating whether to compute p-values by Monte Carlo simulation, in larger than 2×2 tables.

`B` an integer specifying the number of replicates used in the Monte Carlo test.

Details

ORA measures the overlap between the target feature set and a list of most altered molecular features in mat. The most altered molecular features can be selected from the top and or bottom of the molecular readout distribution, by default it is the top 5% positive values. With these, a contingency table is build and a one-tailed Fisher's exact test is computed to determine if a regulator's set of features are over-represented in the selected features from the data. The resulting score, `ora`, is the minus log10 of the obtained p-value.

Value

A long format tibble of the enrichment scores for each source across the samples. Resulting tibble contains the following columns:

1. `statistic`: Indicates which method is associated with which score.
2. `source`: Source nodes of `network`.
3. `condition`: Condition representing each column of `mat`.
4. `score`: Regulatory activity (enrichment score).

See Also

Other decoupleR statistics: `decouple()`, `run_aucell()`, `run_fgsea()`, `run_gsva()`, `run_mdt()`, `run_mlm()`, `run_udt()`, `run_ulm()`, `run_viper()`, `run_wmean()`, `run_wsum()`

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")

mat <- readRDS(file.path(inputs_dir, "mat.rds"))
net <- readRDS(file.path(inputs_dir, "net.rds"))
```

```
run_ora(mat, net, minsize=0)
```

`run_udt` *Univariate Decision Tree (UDT)*

Description

Calculates regulatory activities by using UDT.

Usage

```
run_udt(
  mat,
  network,
  .source = source,
  .target = target,
  .mor = mor,
  .likelihood = likelihood,
  sparse = FALSE,
  center = FALSE,
  na.rm = FALSE,
  min_n = 20,
  seed = 42,
  minsize = 5
)
```

Arguments

`mat`	Matrix to evaluate (e.g. expression matrix). Target nodes in rows and conditions in columns. `rownames(mat)` must have at least one intersection with the elements in `network` `.target` column.
`network`	Tibble or dataframe with edges and it's associated metadata.
`.source`	Column with source nodes.
`.target`	Column with target nodes.
`.mor`	Column with edge mode of regulation (i.e. mor).
`.likelihood`	Deprecated argument. Now it will always be set to 1.
`sparse`	Deprecated parameter.
`center`	Logical value indicating if `mat` must be centered by `base::rowMeans()`.
`na.rm`	Should missing values (including NaN) be omitted from the calculations of `base::rowMeans()`?
`min_n`	An integer for the minimum number of data points in a node that are required for the node to be split further.
`seed`	A single value, interpreted as an integer, or NULL for random number generation.
`minsize`	Integer indicating the minimum number of targets per source.

Details

UDT fits a single regression decision tree for each sample and regulator, where the observed molecular readouts in mat are the response variable and the regulator weights in net are the explanatory one. Target features with no associated weight are set to zero. The obtained feature importance from the fitted model is the activity `udt` of a given regulator.

Value

A long format tibble of the enrichment scores for each source across the samples. Resulting tibble contains the following columns:

1. `statistic`: Indicates which method is associated with which score.
2. `source`: Source nodes of `network`.
3. `condition`: Condition representing each column of `mat`.
4. `score`: Regulatory activity (enrichment score).

See Also

Other decoupleR statistics: `decouple()`, `run_aucell()`, `run_fgsea()`, `run_gsva()`, `run_mdt()`, `run_mlm()`, `run_ora()`, `run_ulm()`, `run_viper()`, `run_wmean()`, `run_wsum()`

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")

mat <- readRDS(file.path(inputs_dir, "mat.rds"))
net <- readRDS(file.path(inputs_dir, "net.rds"))

run_udt(mat, net, minsize=0)
```

---

run_ulm *Univariate Linear Model (ULM)*

Description

Calculates regulatory activities using ULM.

Usage

```
run_ulm(
  mat,
  network,
  .source = source,
  .target = target,
  .mor = mor,
  .likelihood = likelihood,
  sparse = FALSE,
  center = FALSE,
  na.rm = FALSE,
  minsize = 5L
)
```

Arguments

mat	Matrix to evaluate (e.g. expression matrix). Target nodes in rows and conditions in columns. rownames(mat) must have at least one intersection with the elements in network .target column.
network	Tibble or dataframe with edges and it's associated metadata.
.source	Column with source nodes.
.target	Column with target nodes.
.mor	Column with edge mode of regulation (i.e. mor).
.likelihood	Deprecated argument. Now it will always be set to 1.
sparse	Deprecated parameter.
center	Logical value indicating if mat must be centered by base::rowMeans().
na.rm	Should missing values (including NaN) be omitted from the calculations of base::rowMeans()?
minsize	Integer indicating the minimum number of targets per source.

Details

ULM fits a linear model for each sample and regulator, where the observed molecular readouts in mat are the response variable and the regulator weights in net are the explanatory one. Target features with no associated weight are set to zero. The obtained t-value from the fitted model is the activity ulm of a given regulator.

Value

A long format tibble of the enrichment scores for each source across the samples. Resulting tibble contains the following columns:

1. statistic: Indicates which method is associated with which score.
2. source: Source nodes of network.
3. condition: Condition representing each column of mat.
4. score: Regulatory activity (enrichment score).

See Also

Other decoupleR statistics: decouple(), run_aucell(), run_fgsea(), run_gsva(), run_mdt(), run_mlm(), run_ora(), run_udt(), run_viper(), run_wmean(), run_wsum()

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")

mat <- readRDS(file.path(inputs_dir, "mat.rds"))
net <- readRDS(file.path(inputs_dir, "net.rds"))

run_ulm(mat, net, minsize=0)
```

`run_viper` *Virtual Inference of Protein-activity by Enriched Regulon analysis (VIPER)*

Description

Calculates regulatory activities using VIPER.

Usage

```
run_viper(
  mat,
  network,
  .source = source,
  .target = target,
  .mor = mor,
  .likelihood = likelihood,
  verbose = FALSE,
  minsize = 5,
  pleiotropy = TRUE,
  eset.filter = FALSE,
  ...
)
```

Arguments

`mat`	Matrix to evaluate (e.g. expression matrix). Target nodes in rows and conditions in columns. `rownames(mat)` must have at least one intersection with the elements in `network` `.target` column.
`network`	Tibble or dataframe with edges and it's associated metadata.
`.source`	Column with source nodes.
`.target`	Column with target nodes.
`.mor`	Column with edge mode of regulation (i.e. mor).
`.likelihood`	Deprecated argument. Now it will always be set to 1.
`verbose`	Logical, whether progression messages should be printed in the terminal.
`minsize`	Integer indicating the minimum number of targets per source.
`pleiotropy`	Logical, whether correction for pleiotropic regulation should be performed.
`eset.filter`	Logical, whether the dataset should be limited only to the genes represented in the interactome.
`...`	Arguments passed on to `viper::viper` `dnull` Numeric matrix for the null model, usually generated by `nullTtest` `nes` Logical, whether the enrichment score reported should be normalized `method` Character string indicating the method for computing the single samples signature, either scale, rank, mad, ttest or none `bootstraps` Integer indicating the number of bootstraps iterations to perform. Only the scale method is implemented with bootstraps.

`adaptive.size` Logical, whether the weighting scores should be taken into account for computing the regulon size

`pleiotropyArgs` list of 5 numbers for the pleotropy correction indicating: regulators p-value threshold, pleiotropic interaction p-value threshold, minimum number of targets in the overlap between pleiotropic regulators, penalty for the pleiotropic interactions and the method for computing the pleiotropy, either absolute or adaptive

`cores` Integer indicating the number of cores to use (only 1 in Windows-based systems)

Details

VIPER (Alvarez et al., 2016) estimates biological activities by performing a three-tailed enrichment score calculation. For further information check the supplementary information of the decoupler manuscript or the original publication.

Alvarez M.J.et al. (2016) Functional characterization of somatic mutations in cancer using network-based inference of protein activity. Nat. Genet., 48, 838–847.

Value

A long format tibble of the enrichment scores for each source across the samples. Resulting tibble contains the following columns:

1. `statistic`: Indicates which method is associated with which score.
2. `source`: Source nodes of `network`.
3. `condition`: Condition representing each column of `mat`.
4. `score`: Regulatory activity (enrichment score).

See Also

Other decoupleR statistics: `decouple()`, `run_aucell()`, `run_fgsea()`, `run_gsva()`, `run_mdt()`, `run_mlm()`, `run_ora()`, `run_udt()`, `run_ulm()`, `run_wmean()`, `run_wsum()`

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")

mat <- readRDS(file.path(inputs_dir, "mat.rds"))
net <- readRDS(file.path(inputs_dir, "net.rds"))

run_viper(mat, net, minsize=0, verbose = FALSE)
```

---

run_wmean *Weighted Mean (WMEAN)*

Description

Calculates regulatory activities using WMEAN.

Usage

```
run_wmean(
  mat,
  network,
  .source = source,
  .target = target,
  .mor = mor,
  .likelihood = likelihood,
  times = 100,
  seed = 42,
  sparse = TRUE,
  randomize_type = "rows",
  minsize = 5
)
```

Arguments

mat	Matrix to evaluate (e.g. expression matrix). Target nodes in rows and conditions in columns. rownames(mat) must have at least one intersection with the elements in network .target column.
network	Tibble or dataframe with edges and it's associated metadata.
.source	Column with source nodes.
.target	Column with target nodes.
.mor	Column with edge mode of regulation (i.e. mor).
.likelihood	Deprecated argument. Now it will always be set to 1.
times	How many permutations to do?
seed	A single value, interpreted as an integer, or NULL for random number generation.
sparse	Should the matrices used for the calculation be sparse?
randomize_type	How to randomize the expression matrix.
minsize	Integer indicating the minimum number of targets per source.

Details

WMEAN infers regulator activities by first multiplying each target feature by its associated weight which then are summed to an enrichment score wmean. Furthermore, permutations of random target features can be performed to obtain a null distribution that can be used to compute a z-score norm_wmean, or a corrected estimate corr_wmean by multiplying wmean by the minus log10 of the obtained empirical p-value.

Value

A long format tibble of the enrichment scores for each source across the samples. Resulting tibble contains the following columns:

1. statistic: Indicates which method is associated with which score.
2. source: Source nodes of network.
3. condition: Condition representing each column of mat.
4. score: Regulatory activity (enrichment score).
5. p_value: p-value for the score of the method.

See Also

Other decoupleR statistics: decouple(), run_aucell(), run_fgsea(), run_gsva(), run_mdt(), run_mlm(), run_ora(), run_udt(), run_ulm(), run_viper(), run_wsum()

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")

mat <- readRDS(file.path(inputs_dir, "mat.rds"))
net <- readRDS(file.path(inputs_dir, "net.rds"))

run_wmean(mat, net, minsize=0)
```

---

run_wsum *Weighted Sum (WSUM)*

Description

Calculates regulatory activities using WSUM.

Usage

```
run_wsum(
  mat,
  network,
  .source = source,
  .target = target,
  .mor = mor,
  .likelihood = likelihood,
  times = 100,
  seed = 42,
  sparse = TRUE,
  randomize_type = "rows",
  minsize = 5
)
```

Arguments

mat	Matrix to evaluate (e.g. expression matrix). Target nodes in rows and conditions in columns. rownames(mat) must have at least one intersection with the elements in network .target column.
network	Tibble or dataframe with edges and it's associated metadata.
.source	Column with source nodes.
.target	Column with target nodes.
.mor	Column with edge mode of regulation (i.e. mor).
.likelihood	Deprecated argument. Now it will always be set to 1.
times	How many permutations to do?
seed	A single value, interpreted as an integer, or NULL for random number generation.

sparse	Should the matrices used for the calculation be sparse?
randomize_type	How to randomize the expression matrix.
minsize	Integer indicating the minimum number of targets per source.

Details

WSUM infers regulator activities by first multiplying each target feature by its associated weight which then are summed to an enrichment score wsum. Furthermore, permutations of random target features can be performed to obtain a null distribution that can be used to compute a z-score norm_wsum, or a corrected estimate corr_wsum by multiplying wsum by the minus log10 of the obtained empirical p-value.

Value

A long format tibble of the enrichment scores for each source across the samples. Resulting tibble contains the following columns:

1. statistic: Indicates which method is associated with which score.
2. source: Source nodes of network.
3. condition: Condition representing each column of mat.
4. score: Regulatory activity (enrichment score).
5. p_value: p-value for the score of the method.

See Also

Other decoupleR statistics: decouple(), run_aucell(), run_fgsea(), run_gsva(), run_mdt(), run_mlm(), run_ora(), run_udt(), run_ulm(), run_viper(), run_wmean()

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")

mat <- readRDS(file.path(inputs_dir, "mat.rds"))
net <- readRDS(file.path(inputs_dir, "net.rds"))

run_wsum(mat, net, minsize=0)
```

---

show_methods *Show methods*

Description

Prints the methods available in decoupleR. The first column correspond to the function name in decoupleR and the second to the method's full name.

Usage

```
show_methods()
```

Examples

```
show_methods()
```

show_resources *Shows available resources in Omnipath. For more information visit the official website for* Rhrefhttps://omnipathdb.org/Omnipath.

Description

Shows available resources in Omnipath. For more information visit the official website for Omnipath.

Usage

```
show_resources()
```

Examples

```
decoupleR::show_resources()
```

tidyeval *Tidy eval helpers*

Description

- `rlang::sym()` creates a symbol from a string and `syms()` creates a list of symbols from a character vector.
- `enquo()` and `enquos()` delay the execution of one or several function arguments. `enquo()` returns a single quoted expression, which is like a blueprint for the delayed computation. `enquos()` returns a list of such quoted expressions.
- `expr()` quotes a new expression *locally*. It is mostly useful to build new expressions around arguments captured with `enquo()` or `enquos()`: `expr(mean(!!enquo(arg), na.rm = TRUE))`.
- `rlang::as_name()` transforms a quoted variable name into a string. Supplying something else than a quoted variable name is an error.

  That's unlike `rlang::as_label()` which also returns a single string but supports any kind of R object as input, including quoted function calls and vectors. Its purpose is to summarise that object into a single label. That label is often suitable as a default name.

  If you don't know what a quoted expression contains (for instance expressions captured with `enquo()` could be a variable name, a call to a function, or an unquoted constant), then use `as_label()`. If you know you have quoted a simple variable name, or would like to enforce this, use `as_name()`.

To learn more about tidy eval and how to use these tools, visit `https://tidyeval.tidyverse.org` and the Metaprogramming section of Advanced R.

Examples

```
if (FALSE) {
    help("nse-defuse", package = "rlang")
}
```

`%>%` *Pipe operator*

Description

See `magrittr::%>%` for details.

Usage

```
lhs %>% rhs
```

Value

Pipe an object forward into a function or call expression.

Examples

```
c(1, 2, 3) %>% sum()
```

Index

∗ **decoupleR formats**
.decoupler_mat_format, 4
.decoupler_network_format, 4
∗ **decoupleR statistics**
decouple, 7
run_aucell, 19
run_fgsea, 21
run_gsva, 22
run_mdt, 25
run_mlm, 26
run_ora, 28
run_udt, 30
run_ulm, 31
run_viper, 33
run_wmean, 34
run_wsum, 36
∗ **internal**
.decoupler_mat_format, 4
.decoupler_network_format, 4
%>%, 39
decoupleR-package, 3
get_profile_of, 12
pivot_wider_profile, 15
randomize_matrix, 17
tidyeval, 38
.data *(*tidyeval*)*, 38
.decoupler_mat_format, *4*, 4
.decoupler_network_format, *4*, 4
.fit_preprocessing, 5
:= *(*tidyeval*)*, 38
%>%, *39*, 39

abort *(*tidyeval*)*, 38
as_label *(*tidyeval*)*, 38
as_name *(*tidyeval*)*, 38

base::rowMeans(), *5*, *25*, *27*, *30*, *32*

check_corr, 5
complete, *12*
convert_f_defaults, 6

decouple, 7, *19*, *22*, *24*, *26, 27*, *29*, *31, 32*, *34*, *36, 37*
decoupleR *(*decoupleR-package*)*, 3
decoupleR-package, 3
dplyr::full_join(), *12*
dplyr::rename(), *6, 7*
dplyr::select(), *7*

enquo *(*tidyeval*)*, 38
enquo(), *38*
enquos *(*tidyeval*)*, 38
enquos(), *38*
exec *(*tidyeval*)*, 38
expand, *12*
expand(), *12*
expr *(*tidyeval*)*, 38
expr(), *38*
extract_sets, 9

fgsea::fgseaMultilevel, *21*
filt_minsize, 10

GeneSetCollection, *23*
get_collectri, 10
get_dorothea, 11
get_ksn_omnipath, 12
get_profile_of, 12
get_progeny, 13
get_resource, 14
get_toy_data, 14
GSVA::gsva, *23*
GSVA::gsvaParam, *23*
GSVA::ssgseaParam, *23*

intersect_regulons, 15

mat_format *(*.decoupler_mat_format*)*, 4

network_format
(.decoupler_network_format*)*, 4

pivot_wider_profile, 15

quo_is_missing *(*tidyeval*)*, 38
quo_is_null *(*tidyeval*)*, 38

randomize_matrix, 17

rename_net, *9*, 18
replace_na(), *12*
rlang::as_label(), *38*
rlang::as_name(), *38*
rlang::sym(), *38*
run_aucell, *8*, 19, *22*, *24*, *26, 27*, *29*, *31, 32*, *34*, *36, 37*
run_consensus, 20
run_fgsea, *8*, *19*, 21, *24*, *26, 27*, *29*, *31, 32*, *34*, *36, 37*
run_gsva, *8*, *19*, *22*, 22, *26, 27*, *29*, *31, 32*, *34*, *36, 37*
run_mdt, *8*, *19*, *22*, *24*, 25, *27*, *29*, *31, 32*, *34*, *36, 37*
run_mlm, *8*, *19*, *22*, *24*, *26*, 26, *29*, *31, 32*, *34*, *36, 37*
run_ora, *8*, *19*, *22*, *24*, *26, 27*, 28, *31, 32*, *34*, *36, 37*
run_udt, *8*, *19*, *22*, *24*, *26, 27*, *29*, 30, *32*, *34*, *36, 37*
run_ulm, *8*, *19*, *22*, *24*, *26, 27*, *29*, *31*, 31, *34*, *36, 37*
run_viper, *8*, *19*, *22*, *24*, *26, 27*, *29*, *31, 32*, 33, *36, 37*
run_wmean, *8*, *19*, *22*, *24*, *26, 27*, *29*, *31, 32*, *34*, 34, *37*
run_wsum, *8*, *19*, *22*, *24*, *26, 27*, *29*, *31, 32*, *34*, *36*, 36

show_methods, 37
show_resources, 38
stats::fisher.test, *28*
sym *(*tidyeval*)*, 38
syms *(*tidyeval*)*, 38
syms(), *38*

tibble::add_column(), *6*
tidyeval, 38

viper::viper, *33*