Package 'decoupleR'

November 25, 2021

Type Package

Title decoupleR: Inferring biological activities from omics data using a collection of methods

Version 2.0.0

Description Computational methods allow the extraction of mechanistic signatures from omics data based on prior knowledge resources, reducing the dimensionality of the data 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. Using decoupleR, we evaluated the performance of contemporary methods on transcriptomic and phospho-proteomic perturbation experiments.

License GPL-3

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

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

Depends R (>= 4.0)

Imports broom, dplyr, GSVA, magrittr, Matrix, purrr, rlang, speedglm, stats, stringr, tibble, tidyr, tidyselect, viper, withr, RobustRankAggreg, fgsea (>= 1.15.4), AUCell, SummarizedExperiment, rpart, ranger

Suggests BiocStyle, covr, knitr, pkgdown, RefManageR, rmarkdown, roxygen2, sessioninfo, testthat

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 7.1.2

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

git_branch RELEASE_3_14

git_last_commit b00f6bf

git_last_commit_date 2021-10-26

Date/Publication 2021-11-25

Author Pau Badia-i-Mompel [aut] (<https://orcid.org/0000-0002-1004-3923>),
Jesús Vélez [cre, 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 Jesús Vélez <jvelezmagic@gmail.com>

R topics documented:

`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)
)
```

---

convert_to_ *Convert a network to run under the method of interest.*

Description

Convert a long-format network to the suggested standard for the specified run_{statistic}(). If the default parameters are not modified, then the function sets its own null values for those columns.

Usage

```
convert_to_(network)

convert_to_aucell(network, .source, .target)

convert_to_ulm(network, .source, .target, .mor = NULL, .likelihood = NULL)

convert_to_mlm(network, .source, .target, .mor = NULL, .likelihood = NULL)

convert_to_wsum(network, .source, .target, .mor = NULL, .likelihood = NULL)

convert_to_wmean(network, .source, .target, .mor = NULL, .likelihood = NULL)

convert_to_viper(network, .source, .target, .mor = NULL, .likelihood = NULL)

convert_to_gsva(network, .source, .target)

convert_to_ora(network, .source, .target)

convert_to_fgsea(network, .source, .target)
```

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	Column with edge likelihood.

Value

- convert_to_ Return same as input.

- convert_to_aucell() Return a named list of sources with associated targets.
- convert_to_gsva() Return a list of sources with associated targets suitable for GSVA::gsva().
- convert_to_wmean() Return a tibble with four columns: source, target, mor and likelihood.
- convert_to_ora() Return a named list of sources with associated targets.
- convert_to_wsum() Returns a tibble with three columns: source, target and mor.
- convert_to_ulm() Returns a tibble with three columns: source, target and mor.
- convert_to_mlm() Returns a tibble with three columns: source, target and mor.
- convert_to_viper() Return a list of sources with associated targets suitable for viper::viper()

See Also

convert_f_defaults()

Examples

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

network <- readRDS(file.path(inputs_dir, "input-dorothea_genesets.rds"))

convert_to_(network)
convert_to_aucell(network, tf, target)
convert_to_gsva(network, tf, target)
convert_to_wmean(network, tf, target, mor, likelihood)
convert_to_ora(network, tf, target)
convert_to_wsum(network, tf, target, mor)
convert_to_ulm(network, tf, target, mor)
convert_to_mlm(network, tf, target, mor)
convert_to_viper(network, tf, target, mor, likelihood)
```

---

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 = .data$source,
  .target = .data$target,
  statistics = c("udt", "mdt", "aucell", "wmean", "wsum", "ulm", "mlm", "viper",
    "gsva", "ora", "fgsea"),
  args = list(NULL),
  consensus_score = TRUE,
  include_time = FALSE,
  show_toy_call = 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.
`statistics`	Statistical methods to be coupled.
`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. Obtained scores are -log10(p-values).
`include_time`	Should the time per statistic evaluated be informed?
`show_toy_call`	The call of each statistic must be informed?

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, "input-expr_matrix.rds"))
    network <- readRDS(file.path(inputs_dir, "input-dorothea_genesets.rds"))

    decouple(
        mat = mat,
        network = network,
        .source = "tf",
        .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")
        )
    )
}
```

`filter_regulons` *Filter network by size of regulons*

Description

Keep only sources which satisfied the condition min_size >= n <= max_size, where n denotes the number of targets per source.

Usage

```
filter_regulons(network, .source, min_size = 1, max_size = Inf)
```

Arguments

network	Tibble or dataframe with edges and it's associated metadata.
.source	Column with source nodes.
min_size	Minimum number of targets allowed per regulon.
max_size	Maximum number of targets allowed per regulon.

Value

Filtered tibble.

Examples

```
inputs_dir <- system.file("testdata", "inputs", package = "decoupleR")
network <- readRDS(file.path(inputs_dir, "input-dorothea_genesets.rds"))
filter_regulons(network, .source = tf, min_size = 30, max_size = 50)
```

`intersect_regulons` *Intersect network target genes with expression matrix.*

Description

Keep only edges which its target genes belong to the expression 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, "input-expr_matrix.rds"))
network <- readRDS(file.path(inputs_dir, "input-dorothea_genesets.rds"))
intersect_regulons(mat, network, tf, target, minsize=5)
```

`run_aucell` *AUCell*

Description

Calculates regulatory activities using Area Under the Curve (AUC) from AUCell

Usage

```
run_aucell(
  mat,
  network,
  .source = .data$source,
  .target = .data$target,
  aucMaxRank = ceiling(0.05 * nrow(rankings)),
  nproc = 4,
  seed = 42
)
```

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.

Details

This function is a wrapper for the method `AUCell`. It uses the "Area Under the Curve" (AUC) to calculate whether a critical subset of input molecular features is enriched for each sample.

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, "input-expr_matrix.rds"))
network <- readRDS(file.path(inputs_dir, "input-dorothea_genesets.rds"))

run_aucell(mat, network, .source='tf', nproc=1)
```

---

run_consensus *Function to generate a consensus score between methods from the result of decouple*

Description

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

Usage

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

Arguments

df	decouple data frame result
include_time	Should the time per statistic evaluated be informed?

Value

Updated tibble with the computed consensus score between methods

---

run_fgsea *Fast Gene Set Enrichment Analysis (FGSEA)*

Description

Calculates regulatory activities using FGSEA.

Usage

```
run_fgsea(
  mat,
  network,
  .source = .data$source,
  .target = .data$target,
  times = 100,
  nproc = 4,
  seed = 42,
  ...
)
```

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.
...	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") 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

This function is a wrapper for the method fgsea::fgsea.

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, "input-expr_matrix.rds"))
network <- readRDS(file.path(inputs_dir, "input-dorothea_genesets.rds"))

run_fgsea(mat, network, .source='tf', nproc=1)
```

`run_gsva` *Gene Set Variation Analysis (GSVA)*

Description

Calculates regulatory activities using GSVA.

Usage

```
run_gsva(
  mat,
  network,
  .source = .data$source,
  .target = .data$target,
  verbose = FALSE,
  method = "gsva",
  ...
)
```

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).
`...`	Arguments passed on to `GSVA::gsva`

Details

This function is a wrapper for the method `GSVA::gsva()`.

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, "input-expr_matrix.rds"))
network <- readRDS(file.path(inputs_dir, "input-dorothea_genesets.rds"))

run_gsva(mat, network, .source='tf', verbose = FALSE)
```

---

run_mdt *Multivariate Decision Trees (MDT)*

Description

Calculates regulatory activities by fitting multivariate decision trees (MDT) using `ranger::ranger()`.

Usage

```
run_mdt(
  mat,
  network,
  .source = .data$source,
  .target = .data$target,
  .mor = .data$mor,
  .likelihood = .data$likelihood,
  sparse = FALSE,
  center = FALSE,
  na.rm = FALSE,
  trees = 10,
  min_n = 20,
  nproc = 4,
  seed = 42
)
```

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	Column with edge likelihood.
sparse	Logical value indicating if the generated profile matrix should be sparse.
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.

Details

MDT fits a multivariate ensemble of decision trees (random forest) to estimate regulatory activities. MDT transforms a given network into an adjacency matrix, placing sources as columns and targets as rows. The matrix is filled with the associated weights for each interaction. This matrix is used to fit a random forest model to predict the observed molecular readouts per sample. The obtained feature importances from the fitted model are the activities of the regulators.

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, "input-expr_matrix.rds"))
network <- readRDS(file.path(inputs_dir, "input-dorothea_genesets.rds"))

run_mdt(mat, network, .source='tf')
```

---

run_mlm *Multivariate Linear Model (MLM)*

Description

Calculates regulatory activities by fitting multivariate linear models (MLM)

Usage

```
run_mlm(
  mat,
  network,
  .source = .data$source,
  .target = .data$target,
  .mor = .data$mor,
  .likelihood = .data$likelihood,
  sparse = FALSE,
  center = FALSE,
  na.rm = 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`	Column with edge likelihood.
`sparse`	Logical value indicating if the generated profile matrix should be sparse.
`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()`?

Details

MLM fits a multivariate linear model to estimate regulatory activities. MLM transforms a given network into an adjacency matrix, placing sources as columns and targets as rows. The matrix is filled with the associated weights for each interaction. This matrix is used to fit a linear model to predict the observed molecular readouts per sample. The obtained t-values from the fitted model are the activities of the regulators.

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, "input-expr_matrix.rds"))
network <- readRDS(file.path(inputs_dir, "input-dorothea_genesets.rds"))

run_mlm(mat, network, .source='tf')
```

---

run_ora *Over Representation Analysis (ORA)*

Description

Calculates regulatory activities using ORA.

Usage

```
run_ora(
  mat,
  network,
  .source = .data$source,
  .target = .data$target,
  n_up = 300,
  n_bottom = 300,
```

```
  n_background = 20000,
  with_ties = TRUE,
  ...
)
```

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.
...	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

Performs an over-representation analysis using `stats::fisher.test()`. Obtained scores are -log10(p-values).

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, "input-expr_matrix.rds"))
network <- readRDS(file.path(inputs_dir, "input-dorothea_genesets.rds"))

run_ora(mat, network, .source='tf')
```

---

run_udt *Univariate Decision Tree (UDT)*

Description

Calculates regulatory activities by fitting univariate decision trees (UDT) using `rpart::rpart()`.

Usage

```
run_udt(
  mat,
  network,
  .source = .data$source,
  .target = .data$target,
  .mor = .data$mor,
```

```
  .likelihood = .data$likelihood,
  sparse = FALSE,
  center = FALSE,
  na.rm = FALSE,
  min_n = 20,
  seed = 42
)
```

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	Column with edge likelihood.
sparse	Logical value indicating if the generated profile matrix should be sparse.
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.

Details

UDT fits a (univariate) decision tree to estimate regulatory activities. UDT fits a decision tree that predicts the observed molecular readouts using the given weights of a regulator as a single co-variate. The obtained feature importance from the fitted model is the activity of the 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, "input-expr_matrix.rds"))
network <- readRDS(file.path(inputs_dir, "input-dorothea_genesets.rds"))

run_udt(mat, network, .source='tf')
```

---

run_ulm *Univariate Linear Model (ULM)*

Description

Calculates regulatory activities by fitting univariate linear models (ULM).

Usage

```
run_ulm(
  mat,
  network,
  .source = .data$source,
  .target = .data$target,
  .mor = .data$mor,
  .likelihood = .data$likelihood,
  sparse = FALSE,
  center = FALSE,
  na.rm = 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	Column with edge likelihood.
sparse	Logical value indicating if the generated profile matrix should be sparse.
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()`?

Details

ULM fits a (univariate) linear model to estimate regulatory activities. ULM fits a linear model that predicts the observed molecular readouts using the given weights of a regulator as a single covariate. The obtained t-value from the fitted model is the activity of the regulator. This approach was first described in: Improved detection of tumor suppressor events in single-cell RNA-Seq data.

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, "input-expr_matrix.rds"))
network <- readRDS(file.path(inputs_dir, "input-dorothea_genesets.rds"))

run_ulm(mat, network, .source='tf')
```

---

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

Description

Calculates regulatory activities using VIPER.

Usage

```
run_viper(
  mat,
  network,
  .source = .data$source,
  .target = .data$target,
  .mor = .data$mor,
  .likelihood = .data$likelihood,
```

```
  verbose = FALSE,
  minsize = 0,
  pleiotropy = T,
  eset.filter = F,
  ...
)
```

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	Column with edge likelihood.
verbose	Logical, whether progression messages should be printed in the terminal.
minsize	Integer indicating the minimum number of targets allowed per regulon.
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

This function is a wrapper for the method `viper::viper()`.

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, "input-expr_matrix.rds"))
network <- readRDS(file.path(inputs_dir, "input-dorothea_genesets.rds"))

run_viper(mat, network, .source='tf', verbose = FALSE)
```

`run_wmean` *Weighted Mean (WMEAN)*

Description

Calculates regulatory activities by computing the WMEAN.

Usage

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

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	Column with edge likelihood.
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.

Details

Infers activity score for each regulator by weighting the molecular readouts of its targets by their mode of regulations and likelihoods. In addition, it runs permutations to calculate empirical p-values, providing normalized (z-score) and corrected activity (estimate * -log10(pval)) scores. This is represented in the statistic column which will contain three values for each call to run_wmean(); **wmean**, **norm_wmean** and **corr_wmean**.

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, "input-expr_matrix.rds"))
network <- readRDS(file.path(inputs_dir, "input-dorothea_genesets.rds"))

run_wmean(mat, network, .source='tf')
```

run_wsum *Weighted Sum (WSUM)*

Description

Calculates regulatory activities by computing the WSUM

Usage

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

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	Column with edge likelihood.
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.

Details

Infers activity score for each regulator by weighting the molecular readouts of its targets by their mode of regulations and likelihoods. In addition, it runs permutations to calculate empirical p-values, providing normalized (z-score) and corrected activity (estimate * -log10(p-value)) scores. This is represented in the `statistic` column which will contain three values for each call to `run_wsum()`; **wsum**, **norm_wsum** and **corr_wsum**.

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, "input-expr_matrix.rds"))
network <- readRDS(file.path(inputs_dir, "input-dorothea_genesets.rds"))

run_wsum(mat, network, .source='tf')
```

Index