Package 'decoupleR'

October 14, 2021

Type Package

Title Package to decouple gene sets from statistics

Version 1.0.0

Description Transcriptome profiling followed by differential gene expression analysis often leads to lists of genes that are hard to analyze and interpret. Downstream analysis tools can be used to summarize deregulation events into a smaller set of biologically interpretable features. In particular, methods that estimate the activity of transcription factors (TFs) from gene expression are commonly used. It has been shown that the transcriptional targets of a TF yield a much more robust estimation of the TF activity than observing the expression of the TF itself. Consequently, for the estimation of transcription factor activities, a network of transcriptional regulation is required in combination with a statistical algorithm that summarizes the expression of the target genes into a single activity score. Over the years, many different regulatory networks and statistical algorithms have been developed, mostly in a fixed combination of one network and one algorithm. To systematically evaluate both networks and algorithms, we developed decoupleR , an R package that allows users to apply efficiently any combination provided.

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

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 7.1.1

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

git_branch RELEASE_3_13

git_last_commit 2d54fd6

git_last_commit_date 2021-05-19

Date/Publication 2021-10-14

Author Jesús Vélez [cre, aut] (<https://orcid.org/0000-0001-5128-3838>),
Christian H. Holland [aut] (<https://orcid.org/0000-0002-3060-5786>)

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_scira(network, .source, .target, .mor = NULL)

convert_to_pscira(network, .source, .target, .mor = NULL)

convert_to_mean(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)
```

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_gsva() Return a list of regulons suitable for GSVA::gsva().
- convert_to_mean() Return a tibble with four columns: tf, target, mor and likelihood.
- convert_to_ora() Return a named list of regulons; tf with associated targets.
- convert_to_pscira() Returns a tibble with three columns: tf, target and mor.
- convert_to_scira() Returns a tibble with three columns: tf, target and mor.
- convert_to_viper() Return a list of regulons 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_gsva(network, tf, target)
convert_to_mean(network, tf, target, mor, likelihood)
convert_to_ora(network, tf, target)
convert_to_pscira(network, tf, target, mor)
convert_to_scira(network, tf, target, mor)
convert_to_viper(network, tf, target, mor, likelihood)
```

`decouple` *Evaluate multiple statistics with same input data*

Description

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

Usage

```
decouple(
  mat,
  network,
  .source,
  .target,
  statistics,
  args = list(NULL),
  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`).

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 tf across the samples. Resulting tibble contains the following columns:

1. statistic: Indicates which method is associated with which score.
2. tf: Source nodes of network.
3. condition: Condition representing each column of mat.
4. score: Regulatory activity (enrichment score).
5. statistic_time: If requested, internal execution time indicator.
6. ...: Columns of metadata generated by certain statistics.

See Also

Other decoupleR statistics: run_gsva(), run_mean(), run_ora(), run_pscira(), run_scira(), run_viper()

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", "mean", "pscira", "scira", "viper"),
        args = list(
            gsva = list(verbose = FALSE),
            mean = list(.mor = "mor", .likelihood = "likelihood"),
            pscira = list(.mor = "mor"),
            scira = list(.mor = "mor"),
            viper = list(
                .mor = "mor",
                .likelihood = "likelihood",
                verbose = FALSE
            )
        )
    )
}
```

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)
```

run_gsva *GSVA wrapper*

Description

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

Usage

```
run_gsva(mat, network, .source = .data$tf, .target = .data$target, ...)
```

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

Value

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

1. statistic: Indicates which method is associated with which score.
2. tf: 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_mean(), run_ora(), run_pscira(), run_scira(), run_viper()

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, tf, target, verbose = FALSE)
```

---

run_mean *Weighted mean*

Description

Calculate the activity of all regulons in network through the conditions in the mat matrix by calculating the mean over the expression of all genes.

Usage

```
run_mean(
  mat,
  network,
  .source = .data$tf,
  .target = .data$target,
  .mor = .data$mor,
  .likelihood = .data$likelihood,
  times = 2,
  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

run_mean() calculates the activity score, but in addition, it takes advantage of the permutations used to calculate the p-value, to provide the normalized activity score. This is represented in the statistic column which will contain two values for each call to run_mean(); **mean** and **normalized_mean**.

Value

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

1. statistic: Indicates which method is associated with which score.
2. tf: 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 mean method.

See Also

Other decoupleR statistics: decouple(), run_gsva(), run_ora(), run_pscira(), run_scira(), run_viper()

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_mean(mat, network, tf, target, mor, likelihood)
```

---

run_ora *Over Representation Analysis - Fisher Exact Test*

Description

Performs an over-representation analysis using stats::fisher.test().

Usage

```
run_ora(
  mat,
  network,
  .source = .data$tf,
  .target = .data$target,
  n_up = nrow(mat),
  n_bottom = 0,
  n_background = NULL,
  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.

Value

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

1. statistic: Indicates which method is associated with which score.
2. tf: 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_gsva(), run_mean(), run_pscira(), run_scira(), run_viper()

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, tf, target)
```

`run_pscira` *PSCIRA (Permutation Single Cell Inference of Regulatory Activity)*

Description

Calculate the regulatory activity of each tf by multiplying the expression values of its objectives with their corresponding associated profiles for each given condition.The result is equal to the z-score of the found value compared to its null distribution.

Usage

```
run_pscira(
  mat,
  network,
  .source = .data$tf,
  .target = .data$target,
  .mor = .data$mor,
  sparse = TRUE,
  times = 10,
  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).
`sparse`	Logical value indicating if the generated profile matrix should be sparse.
`times`	Number of replications.
`seed`	A single value, interpreted as an integer, or NULL.

Details

Estimation of regulatory activity: A linear regression of the expression profile is performed against the "target profile" of the given TF, where in the target profile, any regulon member is assigned a `+1` for activating interactions and a `-1` for inhibitory interactions. All other genes not members of the TF's regulon are assigned a value o `0`. TF activity is then defined as the t-statistic of this linear regression.

Value

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

1. `statistic`: Indicates which method is associated with which score.
2. `tf`: 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_gsva()`, `run_mean()`, `run_ora()`, `run_scira()`, `run_viper()`

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_pscira(mat, network, tf, target, mor)
```

`run_scira` *SCIRA (Single Cell Inference of Regulatory Activity)*

Description

Calculates TF activity according to Improved detection of tumor suppressor events in single-cell RNA-Seq data .

Usage

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

```
  sparse = FALSE,
  fast = TRUE,
  center = TRUE,
  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).
sparse	Logical value indicating if the generated profile matrix should be sparse.
fast	Logical value indicating if the lineal model must be calculated with speedglm::speedlm.fit() or with base stats::lm().
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

Estimation of regulatory activity: A linear regression of the expression profile is performed against the "target profile" of the given TF, where in the target profile, any regulon member is assigned a +1 for activating interactions and a -1 for inhibitory interactions. All other genes not members of the TF's regulon are assigned a value o 0. TF activity is then defined as the t-statistic of this linear regression.

Value

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

1. statistic: Indicates which method is associated with which score.
2. tf: 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_gsva(), run_mean(), run_ora(), run_pscira(), run_viper()

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_scira(mat, network, tf, target, mor)
```

`run_viper` *VIPER wrapper*

Description

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

Usage

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

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.
`...`	Arguments passed on to `viper::viper` `dnull` Numeric matrix for the null model, usually generated by `nullTtest` `pleiotropy` Logical, whether correction for pleiotropic regulation should be performed `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

`eset.filter` Logical, whether the dataset should be limited only to the genes represented in the interactome #' @param mvws Number or vector indicating either the exponent score for the metaViper weights, or the inflection point and trend for the sigmoid function describing the weights in metaViper

`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)

`verbose` Logical, whether progression messages should be printed in the terminal

Value

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

1. `statistic`: Indicates which method is associated with which score.
2. `tf`: 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_gsva()`, `run_mean()`, `run_ora()`, `run_pscira()`, `run_scira()`

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, tf, target, mor, likelihood, verbose = FALSE)
```

Index