Package 'matric'

March 22, 2023

Type Package

Title Metrics for Similarity Matrices

Version 0.2.0

Description Calculate quality metrics for readouts from high-throughput profiling experiments.

License BSD_3_clause + file LICENSE

Encoding UTF-8

LazyData true

Imports arrow, dplyr, magrittr, tibble, tidyr, readr, jsonlite, glue, purrr, ggplot2, stringr, rlang (>= 0.4.11), yardstick, foreach, logger, furrr, future

RoxygenNote 7.2.3

Suggests testthat (>= 3.0.0), covr, knitr, markdown, rmarkdown, roxygen2

Config/testthat/edition 3

URL https://github.com/cytomining/matric

BugReports https://github.com/cytomining/matric/issues

Depends R (>= 4.1.0)

VignetteBuilder knitr

NeedsCompilation no

Author Shantanu Singh [aut, cre, cph], Gregory Way [aut]

Maintainer Shantanu Singh <shsingh@broadinstitute.org>

Repository CRAN

Date/Publication 2023-03-22 08:50:02 UTC

R topics documented:

`bin` *Bin values into increasingly wider bins*

Description

Bin values into increasingly wider bins

Usage

```
bin(x)
```

Arguments

x	values

Value

rebinned values

`cellhealth` *Cell Painting dataset of CRISPR perturbations.*

Description

Cell Painting dataset of CRISPR perturbations from doi: 10.1091/mbc.E20120784

Usage

```
cellhealth
```

Format

A data frame with 198 rows and 8 variables:

Metadata_Plate Plate id

Metadata_Well Well id

Metadata_cell_line Cell line id

Metadata_gene_name Gene name

Metadata_pert_name CRISPR guide name

Cells_AreaShape_Compactness compactness measure

Cells_AreaShape_Extent extent measure

Cells_AreaShape_Zernike_0_0 shape measure

Source

https://github.com/broadinstitute/cell-health/

cellhealthmetrics *Quality metrics for Cell Painting dataset of CRISPR perturbations.*

Description

Quality metrics for Cell Painting dataset of CRISPR perturbations from https://www.molbiolcell.org/doi/abs/10.1091/mbc.E20-12-0784

Usage

```
cellhealthmetrics
```

Format

A list of data frames comprising quality metrics at three levels.

Source

https://github.com/broadinstitute/cell-health/

drop_annotation *Drop row annotations.*

Description

`drop_annotation` drops row annotations.

Usage

```
drop_annotation(population, annotation_prefix = "Metadata_")
```

Arguments

`population`	data.frame with annotations (a.k.a. metadata) and observation variables.
`annotation_prefix`	optional character string specifying prefix for annotation columns (e.g. `"Metadata_"` (default)).

Value

data.frame with all columns except row annotations.

Examples

```
suppressMessages(suppressWarnings(library(magrittr)))
population <- tibble::tibble(
  Metadata_group = c(
    "control", "control", "control", "control",
    "experiment", "experiment", "experiment", "experiment"
  ),
  Metadata_batch = c("a", "a", "b", "b", "a", "a", "b", "b"),
  AreaShape_Area = c(10, 12, 15, 16, 8, 8, 7, 7)
)
matric::drop_annotation(population, annotation_prefix = "Metadata_")
```

---

get_annotation *Get row annotations.*

Description

`get_annotation` gets row annotations.

Usage

```
get_annotation(population, annotation_prefix = "Metadata_")
```

Arguments

`population`	data.frame with annotations (a.k.a. metadata) and observation variables.
`annotation_prefix`	optional character string specifying prefix for annotation columns (e.g. `"Metadata_"` (default)).

Value

data.frame with row annotations of the same class as `population`.

Examples

```
suppressMessages(suppressWarnings(library(magrittr)))
population <- tibble::tibble(
  Metadata_group = c(
    "control", "control", "control", "control",
    "experiment", "experiment", "experiment", "experiment"
  ),
  Metadata_batch = c("a", "a", "b", "b", "a", "a", "b", "b"),
  AreaShape_Area = c(10, 12, 15, 16, 8, 8, 7, 7)
)
matric::get_annotation(population, annotation_prefix = "Metadata_")
```

get_p_value *Compute p-value against null distribution*

Description

Compute p-value against null distribution

Usage

```
get_p_value(nulls, m, n, statistic, metric_name = "average_precision")
```

Arguments

nulls	Null distribution data frame
m	Number of positive examples
n	Number of negative examples
statistic	statistic for which to compute p-value
metric_name	name of metric. Only "average_precision" is implemented.

Value

p-value

matric *matric: Metrics for Similarity Matrices*

Description

This is a prototype for testing out ideas for cytominer-eval https://github.com/cytomining/cytominer-eval.

`null_distribution` *Compute null distribution for a set of metrics*

Description

Compute null distribution for a set of metrics

Usage

```
null_distribution(
  metrics,
  background_type,
  level_identifier,
  metric_name = "average_precision",
  n_iterations = 10000,
  random_seed = 42
)
```

Arguments

`metrics`	Metrics data frame, containing at least the column "sim_stat_background_n_{background_type}_{le
`background_type`	Background type. Either "ref" or "non_rep".
`level_identifier`	Level identifier. Either "i" (Level 1_0) or "g" (Level 2_1).
`metric_name`	name of metric. Only "average_precision" is currently implemented.
`n_iterations`	number of iterations for generating the null distribution
`random_seed`	Random seed (default = 42)

Value

Nulls data frame

`null_distribution_helper` *Compute null distribution of metrics*

Description

Compute null distribution of metrics

Usage

```
null_distribution_helper(m, n, nn = 10000, metric_name = "average_precision")
```

Arguments

m	Number of positive examples (= number of replicates - 1)
n	Number of negative examples (= number of controls, or number of non-replicates)
nn	Number of simulations (default = 10000)
metric_name	name of metric. Only "average_precision" is implemented.

Value

Null distribution data frame parametrized by m and n

---

preprocess_data *Preprocess data.* preprocess_data *preprocesses data.*

Description

Preprocess data. preprocess_data preprocesses data.

Usage

```
preprocess_data(population, annotation_prefix = "Metadata_")
```

Arguments

population	data.frame with annotations (a.k.a. metadata) and observation variables.
annotation_prefix	optional character string specifying prefix for annotation columns (e.g. "Metadata_" (default)).

Value

data.frame after preprocessing.

Examples

```
suppressMessages(suppressWarnings(library(magrittr)))
population <- tibble::tibble(
  AreaShape_Area = c(10, 12, 15, 16, 8, 8, 7, 7),
  AreaShape_Compactness = c(10, 12, NA, 16, 8, 8, 7, 7)
)
matric::drop_annotation(population, annotation_prefix = "Metadata_")
```

`sim_annotate` *Annotate a melted similarity matrix.*

Description

`sim_annotate` annotates a melted similarity matrix.

Usage

```
sim_annotate(
  sim_df,
  row_metadata,
  annotation_cols,
  index = "both",
  sim_cols = c("id1", "id2", "sim")
)
```

Arguments

`sim_df`	data.frame with melted similarity matrix.
`row_metadata`	data.frame with row metadata.
`annotation_cols`	character vector specifying annotation columns.
`index`	optional character string specifying whether to annotate left index, right index, or both. This must be one of the strings `"both"` (default), `"left"`, `"right"`.
`sim_cols`	optional character string specifying minimal set of columns for a similarity matrix

Value

Annotated melted similarity matrix of the same class as `sim_df`.

Examples

```
suppressMessages(suppressWarnings(library(magrittr)))
population <- tibble::tibble(
  Metadata_group = sample(c("a", "b"), 4, replace = TRUE),
  Metadata_type = sample(c("x", "y"), 4, replace = TRUE),
  x = rnorm(4),
  y = x + rnorm(4) / 100,
  z = y + rnorm(4) / 1000
)
annotation_cols <- c("Metadata_group")
sim_df <- matric::sim_calculate(population, method = "pearson")
row_metadata <- attr(sim_df, "row_metadata")
matric::sim_annotate(sim_df, row_metadata, annotation_cols)
```

`sim_calculate` *Calculate a melted similarity matrix.*

Description

`sim_calculate` calculates a melted similarity matrix.

Usage

```
sim_calculate(
  population,
  annotation_prefix = "Metadata_",
  strata = NULL,
  method = "pearson",
  lazy = FALSE,
  all_same_cols_rep_or_group = NULL,
  all_same_cols_ref = NULL,
  all_same_cols_rep_ref = NULL,
  reference = NULL,
  ...
)
```

Arguments

`population`	data.frame with annotations (a.k.a. metadata) and observation variables.
`annotation_prefix`	optional character string specifying prefix for annotation columns.
`strata`	optional character vector specifying stratification columns.
`method`	optional character string specifying method for to calculate similarity. This must be one of the strings `"pearson"` (default), `"kendall"`, `"spearman"`, `"euclidean"`, `"cosine"`.
`lazy`	optional boolean specifying whether to lazily evaluate similarity.
`all_same_cols_rep_or_group`	optional character vector specifying columns.
`all_same_cols_ref`	optional character vector specifying columns.
`all_same_cols_rep_ref`	optional character vector specifying columns.
`reference`	optional character string specifying reference.
`...`	arguments passed downstream for parallel processing.

Value

`metric_sim` object, with similarity matrix and related metadata

Examples

```
suppressMessages(suppressWarnings(library(magrittr)))
population <- tibble::tribble(
  ~Metadata_group1, ~Metadata_group2, ~x, ~y, ~z,
  1, 1, -1, 5, -5,
  1, 2, -1.2, 5.1, -5.2,
  2, 1, 0, 6, -4,
  2, 2, 0.3, 6.2, -4.4,
  3, 1, 7, -4, 3,
  3, 2, 7.2, -4.1, 3.7
)
sim_pearson <- matric::sim_calculate(population, method = "pearson")
sim_cosine <- matric::sim_calculate(population, method = "cosine")
sim_euclidean <- matric::sim_calculate(population, method = "euclidean")

sim_pearson %>%
  dplyr::inner_join(sim_cosine,
    by = c("id1", "id2"),
    suffix = c("_pearson", "_cosine")
  ) %>%
  dplyr::inner_join(sim_euclidean %>% dplyr::rename(sim_euclidean = sim),
    by = c("id1", "id2")
  )

sim_cosine <-
  matric::sim_calculate(population,
    strata = "Metadata_group1",
    method = "cosine",
    lazy = TRUE
  )

matric::sim_calculate(population,
  method = "cosine",
  lazy = TRUE,
  all_same_cols_rep_or_group = c("Metadata_group2")
)

matric::sim_calculate(population,
  method = "cosine",
  lazy = TRUE,
  all_same_cols_rep_or_group = c("Metadata_group2"),
  all_same_cols_ref = c("Metadata_group1"),
  reference = data.frame(Metadata_group2 = 2)
)

matric::sim_calculate(population,
  method = "cosine",
  lazy = TRUE,
  all_same_cols_rep_or_group = c("Metadata_group2"),
  all_same_cols_ref = c("Metadata_group1"),
  all_same_cols_rep_ref = c("Metadata_group2"),
  reference = data.frame(Metadata_group2 = 2)
```

```
)
```

`sim_calculate_ij` *Calculate similarities given pairs of rows*

Description

`sim_calculate_ij` calculates similarities given pairs of rows.

Usage

```
sim_calculate_ij(
  population,
  index,
  method = NULL,
  annotation_prefix = "Metadata_",
  ...
)
```

Arguments

`population`	data.frame with annotations (a.k.a. metadata) and observation variables.
`index`	data.frame with at least two columns `id1` and `id2` specifying rows of `population`, and an optional attribute `metric_metadata$method`, which is a character string specifying implemented. Preserve the diagonal entries when constructing `index`
`method`	optional character string specifying method for to calculate similarity. `method`, if specified, overrides `attr(index, "metric_metadata")$method`.
`annotation_prefix`	optional character string specifying prefix for annotation columns.
`...`	arguments passed downstream for parallel processing.

Value

data.frame which is the same as `index`, but with a new column `sim` containing similarities, and with the diagonals filtered out.

Examples

```
suppressMessages(suppressWarnings(library(magrittr)))
population <- tibble::tribble(
  ~Metadata_group, ~x, ~y, ~z,
  1, -1, 5, -5,
  2, 0, 6, -4,
  3, 7, -4, 3,
  4, 14, -8, 6
)
```

```
n <- nrow(population)

index <-
  expand.grid(id1 = seq(n), id2 = seq(n), KEEP.OUT.ATTRS = FALSE)

matric::sim_calculate_ij(population, index, method = "cosine")

attr(index, "metric_metadata") <- list(method = "cosine")

matric::sim_calculate_ij(population, index)
```

`sim_collate` *Collate several subsets of a melted similarity matrix, required for computing metrics.*

Description

`sim_collate` collates several subsets of a melted similarity matrix, required for computing metrics.

Usage

```
sim_collate(
  sim_df,
  all_same_cols_rep,
  annotation_cols,
  any_different_cols_rep = NULL,
  all_different_cols_rep = NULL,
  all_same_cols_ref = NULL,
  all_same_cols_rep_ref = NULL,
  all_same_cols_non_rep = NULL,
  any_different_cols_non_rep = NULL,
  all_different_cols_non_rep = NULL,
  any_different_cols_group = NULL,
  all_same_cols_group = NULL,
  reference = NULL,
  drop_reference = FALSE,
  drop_group = NULL
)
```

Arguments

`sim_df` `metric_sim` object.

`all_same_cols_rep`
optional character vector specifying columns.

`annotation_cols`
character vector specifying which columns from `metadata` to annotate the left index of the filtered `sim_df` with.

any_different_cols_rep
: optional character vector specifying columns.

all_different_cols_rep
: optional character vector specifying columns.

all_same_cols_ref
: optional character vector specifying columns.

all_same_cols_rep_ref
: optional character vector specifying columns.

all_same_cols_non_rep
: optional character vector specifying columns.

any_different_cols_non_rep
: optional character vector specifying columns.

all_different_cols_non_rep
: optional character vector specifying columns.

any_different_cols_group
: optional character vector specifying columns.

all_same_cols_group
: optional character vector specifying columns.

reference
: optional character string specifying reference.

drop_reference
: optional boolean specifying whether to filter (drop) pairs using `reference` on the left index.

drop_group
: optional tbl; rows that match on `drop_group` on the left or right index are dropped.

Details

0. Filter out some rows:

Filter out pairs that match `drop_group` in either right or left indices

1. Similarity to reference:

Fetch similarities between

- (a) all rows (except, optionally those containing `reference`), and
- (b) all rows containing `reference`

Do so only for those (a, b) pairs that

- have *same* values in *all* columns of `all_same_cols_ref`

2. Similarity to replicates (no references):

Fetch similarities between

- (a) all rows except `reference` rows, and
- (b) all rows except `reference` rows (i.e. to each other)

Do so for only those (a, b) pairs that

- have *same* values in *all* columns of `all_same_cols_rep`
- have *different* values in *all* columns of `all_different_cols_rep` (if specified)
- have *different* values in *at least one* column of `any_different_cols_rep` (if specified)

Keep, both, (a, b) and (b, a)

3. Similarity to replicates (only references):

Fetch similarities between

- (a) all rows containing `reference`, and
- (b) all rows containing `reference` (i.e. to each other)

Do so for only those (a, b) pairs that

- have *same* values in *all* columns of `all_same_cols_rep_ref`.

Keep, both, (a, b) and (b, a)

4. Similarity to non-replicates:

Fetch similarities between

- (a) all rows (except, optionally, `reference` rows), and
- (b) all rows except `reference` rows

Do so for only those (a, b) pairs that

- have *same* values in *all* columns of `all_same_cols_non_rep`
- have *different* values in *all* columns `all_different_cols_non_rep`
- have *different* values in *at least one* column of `any_different_cols_non_rep`

Keep, both, (a, b) and (b, a)

5. Similarity to group:

Fetch similarities between

- (a) all rows (except, optionally, `reference` rows), and
- (b) all rows (except, optionally, `reference` rows)

Do so for only those (a, b) pairs that

- have *same* values in *all* columns of `all_same_cols_group`
- have *different* values in *at least one* column of `any_different_cols_group`

Keep, both, (a, b) and (b, a)

Value

`metric_sim` object comprising a filtered `sim_df` with sets of pairs, preserving the same `metric_sim` attributes as `sim_df`.

Examples

```
sim_df <- matric::sim_calculate(matric::cellhealth)

drop_group <-
  data.frame(Metadata_gene_name = "EMPTY")

reference <-
  data.frame(Metadata_gene_name = c("Chr2"))
```

```
all_same_cols_ref <-
  c(
    "Metadata_cell_line",
    "Metadata_Plate"
  )

all_same_cols_rep <-
  c(
    "Metadata_cell_line",
    "Metadata_gene_name",
    "Metadata_pert_name"
  )

all_same_cols_rep_ref <-
  c(
    "Metadata_cell_line",
    "Metadata_gene_name",
    "Metadata_pert_name",
    "Metadata_Plate"
  )

any_different_cols_non_rep <-
  c(
    "Metadata_cell_line",
    "Metadata_gene_name",
    "Metadata_pert_name"
  )

all_same_cols_non_rep <-
  c(
    "Metadata_cell_line",
    "Metadata_Plate"
  )

all_different_cols_non_rep <-
  c("Metadata_gene_name")

all_same_cols_group <-
  c(
    "Metadata_cell_line",
    "Metadata_gene_name"
  )

any_different_cols_group <-
  c(
    "Metadata_cell_line",
    "Metadata_gene_name",
    "Metadata_pert_name"
  )

annotation_cols <-
  c(
    "Metadata_cell_line",
```

```
    "Metadata_gene_name",
    "Metadata_pert_name"
  )

collated_sim <-
  matric::sim_collate(
    sim_df,
    reference = reference,
    all_same_cols_rep = all_same_cols_rep,
    all_same_cols_rep_ref = all_same_cols_rep_ref,
    all_same_cols_ref = all_same_cols_ref,
    any_different_cols_non_rep = any_different_cols_non_rep,
    all_same_cols_non_rep = all_same_cols_non_rep,
    all_different_cols_non_rep = all_different_cols_non_rep,
    any_different_cols_group = any_different_cols_group,
    all_same_cols_group = all_same_cols_group,
    annotation_cols = annotation_cols,
    drop_group = drop_group
  )

head(collated_sim)

collated_sim %>%
  dplyr::group_by(type) %>%
  dplyr::tally()
```

`sim_filter_all_same` *Filter a melted similarity matrix to keep pairs with the same values in specific columns.*

Description

`sim_filter_all_same` filters a melted similarity matrix to keep pairs with the same values in specific columns.

Usage

```
sim_filter_all_same(
  sim_df,
  row_metadata,
  all_same_cols,
  annotation_cols = NULL,
  include_group_tag = FALSE,
  drop_lower = FALSE,
  sim_cols = c("id1", "id2", "sim")
)
```

Arguments

sim_df	data.frame with melted similarity matrix.
row_metadata	data.frame with row metadata.
all_same_cols	character vector specifying columns.
annotation_cols	optional character vector specifying which columns from metadata to annotate the left index of the filtered sim_df with.
include_group_tag	optional boolean specifying whether to include an identifier for the pairs using the values in the all_same_cols columns.
drop_lower	optional boolean specifying whether to drop the pairs where the first index is smaller than the second index. This is equivalent to dropping the lower triangular of sim_df.
sim_cols	optional character string specifying minimal set of columns for a similarity matrix

Value

Filtered sim_df as a data.frame, where only pairs with the same values in all_same_cols columns are kept. Rows are annotated based on the first index, if specified.

Examples

```
suppressMessages(suppressWarnings(library(magrittr)))
n <- 5
population <- tibble::tibble(
  Metadata_group = sample(c("a", "b"), n, replace = TRUE),
  Metadata_type = sample(c("x", "y"), n, replace = TRUE),
  x = rnorm(n),
  y = x + rnorm(n) / 100,
  z = y + rnorm(n) / 1000
)
annotation_cols <- c("Metadata_group", "Metadata_type")
sim_df <- matric::sim_calculate(population, method = "pearson")
row_metadata <- attr(sim_df, "row_metadata")
sim_df <- matric::sim_annotate(sim_df, row_metadata, annotation_cols)
all_same_cols <- c("Metadata_group")
include_group_tag <- TRUE
drop_lower <- FALSE
matric::sim_filter_all_same(
  sim_df,
  row_metadata,
  all_same_cols,
  annotation_cols,
  include_group_tag,
  drop_lower
)
```

`sim_filter_all_same_keep_some` *Filter a melted similarity matrix to keep pairs with the same values in specific columns, and keep only some of these pairs.*

Description

`sim_filter_all_same` filters a melted similarity matrix to keep pairs with the same values in specific columns, keeping only some of these pairs.

Usage

```
sim_filter_all_same_keep_some(
  sim_df,
  row_metadata,
  all_same_cols,
  filter_keep_right,
  annotation_cols = NULL,
  drop_reference = TRUE,
  sim_cols = c("id1", "id2", "sim")
)
```

Arguments

`sim_df`	data.frame with melted similarity matrix.
`row_metadata`	data.frame with row metadata.
`all_same_cols`	character vector specifying columns.
`filter_keep_right`	data.frame of metadata specifying which rows to keep on the right index.
`annotation_cols`	optional character vector specifying which columns from `metadata` to annotate the left index of the filtered `sim_df` with.
`drop_reference`	optional boolean specifying whether to filter (drop) pairs using `filter_keep_right` on the left index.
`sim_cols`	optional character string specifying minimal set of columns for a similarity matrix

Value

Filtered `sim_df` as a data.frame, where only pairs with the same values in `all_same_cols` columns are kept, with further filtering using `filter_keep_right`.Rows are annotated based on the first index, if specified.

Examples

```
suppressMessages(suppressWarnings(library(magrittr)))
n <- 20
population <- tibble::tibble(
  Metadata_group = sample(c("a", "b"), n, replace = TRUE),
  Metadata_type = sample(c("x", "y"), n, replace = TRUE),
  x = rnorm(n),
  y = x + rnorm(n) / 100,
  z = y + rnorm(n) / 1000
)
annotation_cols <- c("Metadata_group", "Metadata_type")
sim_df <- matric::sim_calculate(population, method = "pearson")
row_metadata <- attr(sim_df, "row_metadata")
sim_df <- matric::sim_annotate(sim_df, row_metadata, annotation_cols)
all_same_cols <- c("Metadata_group")
filter_keep_right <-
  tibble::tibble(Metadata_group = "a", Metadata_type = "x")
drop_reference <- FALSE
matric::sim_filter_all_same_keep_some(
  sim_df,
  row_metadata,
  all_same_cols,
  filter_keep_right,
  annotation_cols,
  drop_reference
)
```

---

sim_filter_keep_or_drop_some

Filter a melted similarity matrix to remove or keep specified rows.

Description

sim_filter_keep_or_drop_some filters a melted similarity matrix to remove or keep specified rows.

Usage

```
sim_filter_keep_or_drop_some(
  sim_df,
  row_metadata,
  filter_keep = NULL,
  filter_drop = NULL,
  filter_side = NULL
)
```

Arguments

sim_df	data.frame with melted similarity matrix.
row_metadata	data.frame with row metadata.
filter_keep	optional data.frame of metadata specifying which rows to keep.
filter_drop	optional data.frame of metadata specifying which rows to drop.
filter_side	character string specifying which index to filter on. This must be one of the strings "left" or "right".

Value

Filtered sim_df as a data.frame, with some rows kept and some rows dropped. No filters applied if both filter_keep and filter_drop are NULL.

Examples

```
suppressMessages(suppressWarnings(library(magrittr)))
population <- tibble::tibble(
  Metadata_group = sample(c("a", "b"), 4, replace = TRUE),
  Metadata_type = sample(c("x", "y"), 4, replace = TRUE),
  x = rnorm(4),
  y = x + rnorm(4) / 100,
  z = y + rnorm(4) / 1000
)
annotation_cols <- c("Metadata_group", "Metadata_type")
sim_df <- matric::sim_calculate(population, method = "pearson")
row_metadata <- attr(sim_df, "row_metadata")
sim_df <- matric::sim_annotate(sim_df, row_metadata, annotation_cols)
filter_keep <- tibble::tibble(Metadata_group = "a", Metadata_type = "x")
filter_drop <- tibble::tibble(Metadata_group = "a", Metadata_type = "x")
matric::sim_filter_keep_or_drop_some(sim_df, row_metadata,
  filter_keep = filter_keep, filter_side = "left"
)
matric::sim_filter_keep_or_drop_some(sim_df, row_metadata,
  filter_drop = filter_drop, filter_side = "left"
)
```

---

sim_filter_some_different_drop_some

Filter a melted similarity matrix to keep pairs with the same values in specific columns, and other constraints.

Description

sim_filter_some_different_drop_some filters a melted similarity matrix to keep pairs with the same values in specific columns, and other constraints.

Usage

```
sim_filter_some_different_drop_some(
  sim_df,
  row_metadata,
  any_different_cols,
  all_same_cols = NULL,
  all_different_cols = NULL,
  filter_drop_left = NULL,
  filter_drop_right = NULL,
  annotation_cols = NULL,
  sim_cols = c("id1", "id2", "sim")
)
```

Arguments

`sim_df`	data.frame with melted similarity matrix.
`row_metadata`	data.frame with row metadata.
`any_different_cols`	character vector specifying columns.
`all_same_cols`	optional character vector specifying columns.
`all_different_cols`	optional character vector specifying columns.
`filter_drop_left`	data.frame of metadata specifying which rows to drop on the left index.
`filter_drop_right`	data.frame of metadata specifying which rows to drop on the right index.
`annotation_cols`	optional character vector specifying which columns from `metadata` to annotate the left index of the filtered `sim_df` with.
`sim_cols`	optional character string specifying minimal set of columns for a similarity matrix

Value

Filtered `sim_df` as a data.frame, keeping only pairs that have

- same values in all columns of `all_same_cols`,
- different values in all columns `all_different_cols`, and
- different values in at least one column of `any_different_cols`,

with further filtering using `filter_drop_left` and `filter_drop_right`. Rows are annotated based on the first index, if specified.

Examples

```
suppressMessages(suppressWarnings(library(magrittr)))
population <- tibble::tibble(
  Metadata_group = sample(c("a", "b"), 4, replace = TRUE),
  Metadata_type1 = sample(c("x", "y"), 4, replace = TRUE),
  Metadata_type2 = sample(c("p", "q"), 4, replace = TRUE),
  x = rnorm(4),
  y = x + rnorm(4) / 100,
  z = y + rnorm(4) / 1000
)
annotation_cols <- c("Metadata_group", "Metadata_type")
sim_df <- matric::sim_calculate(population, method = "pearson")
row_metadata <- attr(sim_df, "row_metadata")
sim_df <- matric::sim_annotate(sim_df, row_metadata, annotation_cols)
all_same_cols <- c("Metadata_group")
all_different_cols <- c("Metadata_type1")
any_different_cols <- c("Metadata_type2")
filter_drop_left <-
  tibble::tibble(Metadata_group = "a", Metadata_type1 = "x")
filter_drop_right <-
  tibble::tibble(Metadata_group = "a", Metadata_type1 = "x")
drop_reference <- FALSE
matric::sim_filter_some_different_drop_some(
  sim_df,
  row_metadata,
  any_different_cols,
  all_same_cols,
  all_different_cols,
  filter_drop_left,
  filter_drop_right,
  annotation_cols
)
```

---

sim_metrics *Compute metrics.*

Description

sim_metrics computes metrics.

Usage

```
sim_metrics(
  collated_sim,
  sim_type_background,
  calculate_grouped = FALSE,
  annotation_prefix = "Metadata_",
  use_furrr = FALSE,
  calculate_pvalue = FALSE,
```

```
  ...
)
```

Arguments

collated_sim	output of sim_collated, which is a data.frame with some attributes.
sim_type_background	character string specifying the background distributions for computing scaled metrics. This must be one of the strings "non_rep" or "ref".
calculate_grouped	optional boolean specifying whether to include grouped metrics.
annotation_prefix	optional character string specifying prefix for annotation columns (e.g. "Metadata_" (default)).
use_furrr	boolean indicating whether to use the furrr library for parallel processing.
calculate_pvalue	optional boolean specifying whether to calculate p-values for the metrics
...	arguments passed down to "sim_metrics_signif"

Value

List of metrics.

Examples

```
suppressMessages(suppressWarnings(library(ggplot2)))

cellhealth_subset <-
  matric::cellhealth %>%
  dplyr::filter(Metadata_cell_line == "A549") %>%
  dplyr::group_by(Metadata_cell_line,
                  Metadata_gene_name,
                  Metadata_pert_name) %>%
  dplyr::slice_sample(n = 3) %>%
  dplyr::ungroup()

sim_df <- matric::sim_calculate(cellhealth_subset)

drop_group <-
  data.frame(Metadata_gene_name = "EMPTY")

reference <-
  data.frame(Metadata_gene_name = c("Chr2"))

all_same_cols_ref <-
  c(
    "Metadata_cell_line",
    "Metadata_Plate"
  )
```

```
all_same_cols_rep <-
  c(
    "Metadata_cell_line",
    "Metadata_gene_name",
    "Metadata_pert_name"
  )

all_same_cols_rep_ref <-
  c(
    "Metadata_cell_line",
    "Metadata_gene_name",
    "Metadata_pert_name",
    "Metadata_Plate"
  )

any_different_cols_non_rep <-
  c(
    "Metadata_cell_line",
    "Metadata_gene_name",
    "Metadata_pert_name"
  )

all_same_cols_non_rep <-
  c(
    "Metadata_cell_line",
    "Metadata_Plate"
  )

all_different_cols_non_rep <-
  c("Metadata_gene_name")

all_same_cols_group <-
  c(
    "Metadata_cell_line",
    "Metadata_gene_name"
  )

any_different_cols_group <-
  c(
    "Metadata_cell_line",
    "Metadata_gene_name",
    "Metadata_pert_name"
  )

annotation_cols <-
  c(
    "Metadata_cell_line",
    "Metadata_gene_name",
    "Metadata_pert_name"
  )

collated_sim <-
```

```
  matric::sim_collate(
    sim_df,
    reference = reference,
    all_same_cols_rep = all_same_cols_rep,
    all_same_cols_rep_ref = all_same_cols_rep_ref,
    all_same_cols_ref = all_same_cols_ref,
    any_different_cols_non_rep = any_different_cols_non_rep,
    all_same_cols_non_rep = all_same_cols_non_rep,
    all_different_cols_non_rep = all_different_cols_non_rep,
    any_different_cols_group = any_different_cols_group,
    all_same_cols_group = all_same_cols_group,
    annotation_cols = annotation_cols,
    drop_group = drop_group
  )

metrics <- matric::sim_metrics(collated_sim, "ref", calculate_grouped = TRUE)

ggplot(
  metrics$level_1_0,
  aes(sim_scaled_mean_ref_i, fill = Metadata_gene_name)
) +
  geom_histogram(binwidth = .1) +
  facet_wrap(~Metadata_cell_line)

ggplot(
  metrics$level_1,
  aes(sim_scaled_mean_ref_i_mean_i, fill = Metadata_gene_name)
) +
  geom_histogram(binwidth = .1) +
  facet_wrap(~Metadata_cell_line)

ggplot(
  metrics$level_2_1,
  aes(sim_scaled_mean_ref_g, fill = Metadata_gene_name)
) +
  geom_histogram(binwidth = .1) +
  facet_wrap(~Metadata_cell_line)
```

`sim_metrics_helper` *Helper function to compute metrics.*

Description

`sim_metrics_helper` helps compute metrics by aggregating and scaling.

Usage

```
sim_metrics_helper(
  collated_sim,
  sim_type_signal,
```

```
  sim_type_background,
  summary_cols,
  annotation_cols,
  identifier = NULL,
  use_furrr = FALSE
)
```

Arguments

collated_sim	output of sim_collated, which is a data.frame with some attributes.
sim_type_signal	character string specifying the type of replication being measured. This must be one of the strings "rep" or "rep_group".
sim_type_background	character string specifying the background distributions for computing scaled metrics. This must be one of the strings "non_rep" or "ref".
summary_cols	character list specifying columns by which to group similarities.
annotation_cols	character list specifying annotation columns.
identifier	character string specifying the identifier to add as a suffix to the columns containing scaled-aggregated metrics.
use_furrr	boolean indicating whether to use the furrr library for parallel processing.

Value

data.frame of metrics.

---

sim_metrics_signif *Report p-values for metrics*

Description

Report p-values for metrics

Usage

```
sim_metrics_signif(
  metrics,
  background_type,
  level_identifier,
  metric_name,
  ...
)
```

Arguments

metrics	Metrics data frame, containing at least two columns "sim_stat_signal_n_{background_type}_{level and "sim_stat_background_n_{background_type}_{level_identifier}".
background_type	Background type. Either "ref" or "non_rep".
level_identifier	Level identifier. Either "i" (Level 1_0) or "g" (Level 2_1).
metric_name	name of metric. Only "average_precision" is currently implemented.
...	arguments passed downstream.

Value

Metrics data frame containing two extra columns: the -log10 p-value and q-value of the specified metric "sim_retrieval_average_precision_{background_type}_{level_identifier}_nlog10pvalue"

---

sim_new *Constructor for* matric_sim *S3 class.*

Description

sim_new creates an object of class matric_sim.

Usage

```
sim_new(x, row_metadata, metric_metadata)
```

Arguments

x	data.frame with similarity matrix.
row_metadata	tbl with row metadata.
metric_metadata	list with metric information

Details

matric_sim is just a tibble with two attributes and at least three columns.

Columns:

- id1 and id2: integers, indicating row ids.
- sim: double, indicating similarity between the rows.

Attributes:

- row_metadata: data.frame of row annotations, with id column.
- metric_metadata: information about the similarity metric.

This is somewhat similar to Biobase::AnnotatedDataFrame.

Value

Object of class `matric_sim`.

`sim_plot` *Plot a melted similarity matrix.*

Description

`sim_plot` plots a melted similarity matrix.

Usage

```
sim_plot(
  sim_df,
  annotation_column,
  calculate_sim_rank = FALSE,
  trim_label = NULL
)
```

Arguments

`sim_df`	data.frame with melted similarity matrix.
`annotation_column`	character string specifying the column in `sim_df` to use to annotate rows and columns.
`calculate_sim_rank`	boolean specifying whether to calculate rank of similarity.
`trim_label`	optional integer specifying the trim length for tick labels.

Value

`ggplot` object of the plot.

Examples

```
suppressMessages(suppressWarnings(library(magrittr)))
population <- tibble::tibble(
  Metadata_group = sample(c("a", "b", "c", "d"), 100, replace = TRUE),
  x1 = rnorm(100),
  x2 = rnorm(100),
  x3 = rnorm(100),
  x4 = rnorm(100),
  x5 = rnorm(100)
)
annotation_cols <- c("Metadata_group", "Metadata_type")
sim_df <- matric::sim_calculate(population, method = "pearson")
row_metadata <- attr(sim_df, "row_metadata")
sim_df <- matric::sim_annotate(sim_df, row_metadata, annotation_cols)
```

```
annotation_column <- "Metadata_group"
matric::sim_plot(sim_df, annotation_column, calculate_sim_rank = TRUE)
```

---

sim_read *Read similarity matrix.*

Description

sim_read reads similarity matrix.

Usage

```
sim_read(input, file_format = "parquet")
```

Arguments

input	character string specifying the input filename or directory.
file_format	character string specify file format. This must be one of csv or parquet(default).

Value

metric_sim object.

Examples

```
suppressMessages(suppressWarnings(library(magrittr)))
population <- tibble::tibble(
  Metadata_group = sample(c("a", "b"), 4, replace = TRUE),
  x = rnorm(4),
  y = x + rnorm(4) / 100,
  z = y + rnorm(4) / 1000
)
tmpdir <- tempdir()
tmpfile_prefix <- file.path(tmpdir, "test")
tmpfile_parquet <- file.path(tmpdir, "test.pqrquet")
sim_df <- sim_calculate(population, method = "pearson")
sim_df %>% sim_write(tmpfile_prefix, file_format = "csv")
sim_df_csv <- sim_read(tmpfile_prefix, file_format = "csv")
sim_df %>% sim_write(tmpfile_parquet)
sim_df_parquet1 <- sim_read(tmpfile_parquet)
sim_df %>% arrow::write_parquet(tmpfile_parquet)
sim_df_parquet2 <- arrow::read_parquet(tmpfile_parquet)
all(sim_df_parquet1 == sim_df_parquet2)
all(sim_df_parquet1 == sim_df_csv)
```

sim_restore *Restorer for* matric_sim *S3 class.*

Description

restore restores the attributes of class matric_sim.

Usage

```
sim_restore(x, x_attributes)
```

Arguments

x	object to preserve.
x_attributes	list of attributes of class matric_sim.

Details

This is a workaround until tibble inheritance improves

https://github.com/tidyverse/tibble/issues/275 https://adv-r.hadley.nz/s3.html#inheritance.

https://github.com/tidyverse/dplyr/issues/5480#issuecomment-682620522 "dplyr is not really ready for extension in this way"

These are some of the dplyr verbs that will necessitate restoration:

- summarise
- group_by

There are likely more!

Value

Object of class matric_sim if x is a valid object of that class

Examples

```
sim_df <-
  matric::sim_new(
    data.frame(id1 = 1, id2 = 2, sim = 1),
    data.frame(id = c(1, 2), Metadata_group = c("a", "b")),
    list(method = "pearson")
  )
sim_df_attr <- attributes(sim_df)
"matric_sim" %in% class(sim_df)
"matric_sim" %in% class(sim_df %>% dplyr::slice(1))
"matric_sim" %in%
  class(
    sim_df %>%
```

```
      dplyr::group_by(id1, id2) %>%
      dplyr::summarize(sim = mean(sim), .groups = "keep")
  )

"matric_sim" %in%
  class(
    sim_df %>%
      dplyr::group_by(id1, id2) %>%
      dplyr::summarize(sim = mean(sim), .groups = "keep") %>%
      matric::sim_restore(sim_df_attr)
  )
```

`sim_validate` *Validator for* `matric_sim` *S3 class.*

Description

`sim_validate` validates that an object is of class class `matric_sim`.

Usage

```
sim_validate(x)
```

Arguments

`x`	object.

Value

Object of class `matric_sim` if `x` is a valid object of that class.

`sim_wider` *Widen a symmetric melted similarity matrix.*

Description

`sim_widen` widens a symmetric melted matrix.

Usage

```
sim_wider(sim_df, annotation_column, primary_key_column)
```

Arguments

sim_df	data.frame with melted similarity matrix.
annotation_column	character string specifying the column in sim_df to use to annotate rows and columns
primary_key_column	character string specifying the column in sim_df to use to uniquely identify rows and columns

Value

data.frame of widened similarity matrix, with some attributes.

Examples

```
suppressMessages(suppressWarnings(library(magrittr)))
population <- tibble::tibble(
  Metadata_group = sample(c("a", "b", "c", "d"), 100, replace = TRUE),
  x1 = rnorm(100),
  x2 = rnorm(100),
  x3 = rnorm(100),
  x4 = rnorm(100),
  x5 = rnorm(100)
)
population$Metadata_id <- seq(nrow(population))
metadata <- matric::get_annotation(population)
annotation_cols <- c("Metadata_group", "Metadata_id")
sim_df <- matric::sim_calculate(population, method = "pearson")
row_metadata <- attr(sim_df, "row_metadata")
sim_df <- matric::sim_annotate(sim_df, row_metadata, annotation_cols)
annotation_column <- "Metadata_group"
primary_key_column <- "Metadata_id"
res <- matric::sim_wider(sim_df, annotation_column, primary_key_column)
res
data.frame(id = rownames(res)) %>% dplyr::inner_join(attr(res, "map"))
```

---

sim_write *Write similarity matrix.*

Description

sim_write writes similarity matrix.

Usage

```
sim_write(sim_df, output, file_format = "parquet")
```

Arguments

sim_df	metric_sim object.
output	character string specifying the output directory or filename.
file_format	character string specify file format. This must be one of csv or parquet(default).

Details

The output format can be either CSV or Parquet.

With the CSV format, the row_metadata and metric_metadata attributes are saved as separate files.

This is not required for Parquet because it saves the attributes as well.

Value

No return value, called for side effects

Examples

```
suppressMessages(suppressWarnings(library(magrittr)))
population <- tibble::tibble(
  Metadata_group = sample(c("a", "b"), 4, replace = TRUE),
  x = rnorm(4),
  y = x + rnorm(4) / 100,
  z = y + rnorm(4) / 1000
)
tmpdir <- tempdir()
tmpfile_prefix <- file.path(tmpdir, "test")
sim_df <- matric::sim_calculate(population, method = "pearson")
sim_df %>% matric::sim_write(tmpfile_prefix, file_format = "csv")
readr::read_csv(file.path(tmpfile_prefix, "test.csv"))
readr::read_csv(file.path(tmpfile_prefix, "test_metadata.csv"))
jsonlite::read_json(file.path(tmpfile_prefix, "test_metadata.json"))
sim_df %>% matric::sim_write(paste0(tmpfile_prefix, ".parquet"))
sim_df_in <- arrow::read_parquet(paste0(tmpfile_prefix, ".parquet"))
attr(sim_df_in, "row_metadata")
attr(sim_df_in, "metric_metadata")
```

---

sparse_pairwise *Compute similarity between pairs of rows of a matrix*

Description

sparse_pairwise computes similarity between pairs of rows of a matrix.

Usage

```
sparse_pairwise(X, id1, id2, pairwise_function, use_furrr = FALSE)
```

Arguments

X	matrix
id1	vector of integers specifying the list of rows of X (first set)
id2	vector of integers specifying the list of rows of X, (second set), same length as id1.
pairwise_function	function that takes a matrix and a pair of indices specifying rows of the matrix, and computes an operation of each pair of rows
use_furrr	boolean indicating whether to use the furrr library for parallel processing.

Value

data.frame with the same number of rows as the length of id1 (and id2) containing the similarity between the pairs of rows of X. sim[i] == pairwise_function(X[id1[i],], X[id2[i],]).

---

sparse_similarity *Compute similarity between pairs of rows of a matrix*

Description

cosine_sparse computes cosine similarity between pairs of rows of a matrix. pearson_sparse computes pearson similarity between pairs of rows of a matrix.

Usage

```
cosine_sparse(X, id1, id2, ...)

pearson_sparse(X, id1, id2, ...)
```

Arguments

X	matrix
id1	vector of integers specifying the list of rows of X (first set)
id2	vector of integers specifying the list of rows of X, (second set), same length as id1.
...	arguments passed downstream for parallel processing.

Value

data.frame with the same number of rows as the length of id1 (and id2) containing the similarity between the pairs of rows of X. sim[i] == similarity(X[id1[i],], X[id2[i],]).

Examples

```
set.seed(42)
X <- matrix(rnorm(5 * 3), 5, 3)

id1 <- c(1, 3)
id2 <- c(5, 4)

s1 <- matric::cosine_sparse(X, id1, id2) %>% dplyr::arrange(id1, id2)

Xn <- X / sqrt(rowSums(X * X))

n_rows <- nrow(Xn)

s2 <-
  expand.grid(
    id1 = seq(n_rows),
    id2 = seq(n_rows),
    KEEP.OUT.ATTRS = FALSE
  ) %>%
  dplyr::mutate(sim = as.vector(tcrossprod(Xn))) %>%
  dplyr::inner_join(s1 %>% dplyr::select(id1, id2)) %>%
  dplyr::arrange(id1, id2)

s1

all.equal(s1, s2)

Xm <- X - rowMeans(X)
s3 <- matric::cosine_sparse(Xm, id1, id2) %>% dplyr::arrange(id1, id2)
s4 <- matric::pearson_sparse(X, id1, id2) %>% dplyr::arrange(id1, id2)

all.equal(s3, s4)
```

---

tcrossprod_ij *Compute cross product between two sets of rows of a matrix.*

Description

`tcrossprod_ij` computes cross product between two sets of rows of a matrix.

Usage

```
tcrossprod_ij(X, id1, id2)
```

Arguments

`X`	matrix
`id1`	vector of integers specifying the list of rows of `X` (first set)

id2 vector of integers specifying the list of rows of X, (second set), same length as id1.

Value

matrix containing the cross product of X[id1,] and X[id2,].

Examples

```
set.seed(42)
X <- matrix(rnorm(5 * 3), 5, 3)

id1 <- c(1, 3)
id2 <- c(5, 4)

(s1 <- matric::tcrossprod_ij(X, id1, id2))

(s2 <- tcrossprod(X)[id1, id2])

all.equal(s1, s2)
```

Index