Package: cytominer (via r-universe)

October 14, 2024

Encoding UTF-8

Type Package

Title Methods for Image-Based Cell Profiling

Version 0.2.2.9000

Description `cytominer` is a suite of common functions used to process high-dimensional readouts from image-based cell profiling experiments.

Depends R (>= 3.3.0)

License BSD_3_clause + file LICENSE

LazyData TRUE

Imports caret (>= 6.0.76), doParallel (>= 1.0.10), dplyr (>= 0.8.5), foreach (>= 1.4.3), futile.logger (>= 1.4.3), magrittr (>= 1.5), Matrix (>= 1.2), purrr (>= 0.3.3), rlang (>= 0.4.5), tibble (>= 2.1.3), tidyr (>= 1.0.2), glue

Suggests DBI (>= 0.7), dbplyr (>= 1.4.2), knitr (>= 1.17), lazyeval (>= 0.2.0), readr (>= 1.1.1), rmarkdown (>= 1.6), RSQLite (>= 2.0), stringr (>= 1.2.0), testthat (>= 1.0.2)

VignetteBuilder knitr

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

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

RoxygenNote 7.1.1

Repository https://cytomining.r-universe.dev

RemoteUrl https://github.com/cytomining/cytominer

RemoteRef HEAD

RemoteSha d911eb5bc152e0fc8e382ee0492356fd939d2488

Contents

`aggregate` *Aggregate data based on given grouping.*

Description

`aggregate` aggregates data based on the specified aggregation method.

Usage

```
aggregate(
  population,
  variables,
  strata,
  operation = "mean",
  univariate = TRUE,
  ...
)
```

Arguments

`population`	tbl with grouping (metadata) and observation variables.
`variables`	character vector specifying observation variables.
`strata`	character vector specifying grouping variables for aggregation.
`operation`	optional character string specifying method for aggregation, e.g. `"mean"`, `"median"`, `"mean+sd"`. A sequence can comprise only of univariate functions.
`univariate`	boolean specifying whether the aggregation function is univariate or multivariate.
`...`	optional arguments passed to aggregation operation

Value

aggregated data of the same class as `population`.

Examples

```
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"),
  Area = c(10, 12, 15, 16, 8, 8, 7, 7),
  Intensity = c(3, -3, 35, -3, 3, 0, 9, -7)
)
variables <- c("Area", "Intensity")
strata <- c("Metadata_group", "Metadata_batch")
aggregate(population, variables, strata, operation = "mean")
aggregate(population, variables, strata, operation = "mean+sd")
aggregate(population, variables, strata, operation = "median")
aggregate(population, variables, strata, operation = "covariance", univariate = FALSE)
```

`correlation_threshold` *Remove redundant variables.*

Description

`correlation_threshold` returns list of variables such that no two variables have a correlation greater than a specified threshold.

Usage

```
correlation_threshold(variables, sample, cutoff = 0.9, method = "pearson")
```

Arguments

`variables`	character vector specifying observation variables.
`sample`	tbl containing sample used to estimate parameters.
`cutoff`	threshold between [0,1] that defines the minimum correlation of a selected feature.
`method`	optional character string specifying method for calculating correlation. This must be one of the strings `"pearson"` (default), `"kendall"`, `"spearman"`.

Details

`correlation_threshold` is a wrapper for `caret::findCorrelation`.

Value

character vector specifying observation variables to be excluded.

Examples

```
suppressMessages(suppressWarnings(library(magrittr)))
sample <- tibble::tibble(
  x = rnorm(30),
  y = rnorm(30) / 1000
)

sample %<>% dplyr::mutate(z = x + rnorm(30) / 10)
variables <- c("x", "y", "z")

head(sample)
cor(sample)

# `x` and `z` are highly correlated; one of them will be removed

correlation_threshold(variables, sample)
```

`count_na_rows` *Count the number of* `NA`*s per variable.*

Description

`count_na_rows` counts the number of `NA`s per variable.

Usage

```
count_na_rows(population, variables)
```

Arguments

`population`	tbl with grouping (metadata) and observation variables.
`variables`	character vector specifying observation variables.

Value

data frame with frequency of `NA`s per variable.

Examples

```
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),
  AreaShape_length = c(2, 3, NA, NA, 4, 5, 1, 5)
)
variables <- c("AreaShape_Area", "AreaShape_length")
count_na_rows(population, variables)
```

`covariance` *Compute covariance matrix and vectorize.*

Description

`covariance` computes the covariance matrix and vectorize it.

Usage

```
covariance(population, variables)
```

Arguments

`population`	tbl with grouping (metadata) and observation variables.
`variables`	character vector specifying observation variables.

Value

data frame of 1 row comprising vectorized covariance matrix.

Examples

```
population <- tibble::tibble(
  x = rnorm(30),
  y = rnorm(30),
  z = rnorm(30)
)

variables <- c("x", "y")

covariance(population, variables)
```

drop_na_columns *Remove variables with NA values.*

Description

`drop_na_columns` returns list of variables which have greater than a specified threshold number of `NA`s.

Usage

```
drop_na_columns(population, variables, cutoff = 0.05)
```

Arguments

`population`	tbl with grouping (metadata) and observation variables.
`variables`	character vector specifying observation variables.
`cutoff`	threshold between [0,1]. Variables with an `NA` frequency > `cutoff` are returned.

Value

character vector specifying observation variables to be excluded.

Examples

```
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),
  AreaShape_Length = c(2, 3, NA, NA, 4, 5, 1, 5)
)
variables <- c("AreaShape_Area", "AreaShape_Length")
drop_na_columns(population, variables)
```

`drop_na_rows` *Drop rows that are* `NA` *in all specified variables.*

Description

`drop_na_rows` drops rows that are `NA` in all specified variables.

Usage

```
drop_na_rows(population, variables)
```

Arguments

`population`	tbl with grouping (metadata) and observation variables.
`variables`	character vector specifying observation variables.

Value

`population` without rows that have `NA` in all specified variables.

Examples

```
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, NA, 16, 8, 8, 7, 7),
  AreaShape_Length = c(2, 3, NA, NA, 4, 5, 1, 5)
)
variables <- c("AreaShape_Area", "AreaShape_Length")
drop_na_rows(population, variables)
```

`extract_subpopulations` *Extract subpopulations.*

Description

`extract_subpopulations` identifies clusters in the reference and population sets and reports the frequency of points in each cluster for the two sets.

Usage

```
extract_subpopulations(population, reference, variables, k)
```

Arguments

`population`	tbl with grouping (metadata) and observation variables.
`reference`	tbl with grouping (metadata) and observation variables. Columns of `population` and `reference` should be identical.
`variables`	character vector specifying observation variables.
`k`	scalar specifying number of clusters.

Value

list containing clusters centers (`subpop_centers`), two normalized histograms specifying frequency of each clusters in population and reference (`subpop_profiles`), and cluster prediction and distance to the predicted cluster for all input data (`population_clusters` and `reference_clusters`).

Examples

```
data <- tibble::tibble(
  Metadata_group = c(
    "control", "control", "control", "control",
    "experiment", "experiment", "experiment", "experiment"
  ),
  AreaShape_Area = c(10, 12, NA, 16, 8, 8, 7, 7),
  AreaShape_Length = c(2, 3, NA, NA, 4, 5, 1, 5)
)
variables <- c("AreaShape_Area", "AreaShape_Length")
population <- dplyr::filter(data, Metadata_group == "experiment")
reference <- dplyr::filter(data, Metadata_group == "control")
extract_subpopulations(
  population = population,
  reference = reference,
  variables = variables,
  k = 3
)
```

`find_significant_pcs` *Find significant PC's given the eigenvalues.*

Description

`find_significant_pcs` finds significant PC's given the eigenvalues.

Usage

```
find_significant_pcs(S, method = "outlier", n = NULL, d = NULL)
```

Arguments

`S`	numeric vector with eigenvalues of covariance matrix, sorted in descending order.
`method`	optional string specifying method to estimate number of significant PCs
`n`	optional integer specifying number of rows in the data matrix. Default is `NULL`
`d`	optional integer specifying number of columns in the data matrix. Default is `NULL`

Value

number of significant PCs

`generalized_log` *Generalized log transform data.*

Description

`generalized_log` transforms specified observation variables using $x = log((x + sqrt(x^2 + offset^2))/2)$.

Usage

```
generalized_log(population, variables, offset = 1)
```

Arguments

`population`	tbl with grouping (metadata) and observation variables.
`variables`	character vector specifying observation variables.
`offset`	optional offset parameter for the transformation.

Value

transformed data of the same class as `population`.

Examples

```
population <- tibble::tibble(
  Metadata_Well = c("A01", "A02", "B01", "B02"),
  Intensity_DNA = c(8, 20, 12, 32)
)
variables <- c("Intensity_DNA")
generalized_log(population, variables)
```

`generate_component_matrix` *A sparse matrix for sparse random projection.*

Description

`generate_component_matrix` generates the sparse random component matrix for performing sparse random projection. If `density` is the density of the sparse matrix and `n_components` is the size of the projected space, the elements of the random matrix are drawn from

Usage

```
generate_component_matrix(n_features, n_components, density)
```

Arguments

`n_features`	the dimensionality of the original space.
`n_components`	the dimensionality of the projected space.
`density`	the density of the sparse random matrix.

Details

`-sqrt(1 / (density * n_components))` with probability `density / 2` `0` with probability `1 - density` `sqrt(1 / (density * n_components))` with probability `density / 2`

Value

A sparse random matrix of size `(n_features, n_components)`.

Examples

```
M <- generate_component_matrix(500, 100, 0.3)
M[1:10, 1:10]
```

`husk` *Husk data.*

Description

husk detects unwanted variation in the sample and removes it from the population.

Usage

```
husk(
  population,
  variables,
  sample,
  remove_outliers = TRUE,
  epsilon = 1e-06,
  remove_signal = TRUE,
  flatten_noise = TRUE
)
```

Arguments

`population`	tbl with grouping (metadata) and observation variables.
`variables`	character vector specifying observation variables.
`sample`	tbl containing sample that is used by the method to estimate husking parameters. `sample` has same structure as `population`. Typically, `sample` corresponds to controls in the experiment.
`remove_outliers`	optional boolean specifying whether to remove outliers. Default is `TRUE`.
`epsilon`	optional parameter used in husking to offset eigenvalues to avoid division by zero. Default is `1`.
`remove_signal`	optional boolean specifying whether to husk the signal instead of only scaling it down. Default is `TRUE`.
`flatten_noise`	optional boolean specifying whether to flatten the noise instead of scaling it up. Default is `TRUE`. The parameter is ignored if `remove_signal` is `FALSE`.

Value

transformed data of the same class as `population`.

Examples

```
population <- tibble::tibble(
  Metadata_pert_name = c(NA, NA, NA, NA),
  Metadata_Well = c("A01", "A02", "B01", "B02"),
  Intensity_DNA = c(10, 20, 12, 32),
  Granularity_DNA = c(22, 20, NA, 32),
  Texture_DNA = c(5, 2, 43, 13)
)
variables <- c("Intensity_DNA", "Texture_DNA")
husk(population, variables, population, epsilon = 1, remove_signal = TRUE)
husk(population, variables, population, epsilon = 1e-5, remove_signal = TRUE)
husk(population, variables, population, epsilon = 1, remove_signal = FALSE)
husk(population, variables, population, epsilon = 1e-5, remove_signal = FALSE)
```

mark_outlier_rows *Mark outlier rows.*

Description

`mark_outlier_rows` drops outlier rows.

Usage

```
mark_outlier_rows(
  population,
  variables,
  sample,
  method = "svd+iqr",
  outlier_col = "is_outlier",
  ...
)
```

Arguments

`population`	tbl with grouping (metadata) and observation variables.
`variables`	character vector specifying observation variables.
`sample`	tbl containing sample that is used by outlier removal methods to estimate parameters. `sample` has same structure as `population`. Typically, `sample` corresponds to controls in the experiment.
`method`	optional character string specifying method for outlier removal. There is currently only one option (`"svd_iqr"`).
`outlier_col`	optional character string specifying the name for the column that will indicate outliers (in the output). Default `"is_outlier"`.
`...`	arguments passed to outlier removal method.

Value

`population` with an extra column `is_outlier`.

Examples

```
suppressMessages(suppressWarnings(library(magrittr)))
population <- tibble::tibble(
  Metadata_group = sample(c("a", "b"), 100, replace = TRUE),
  Metadata_type = sample(c("control", "trt"), 100, replace = TRUE),
  AreaShape_Area = c(rnorm(98), 20, 30),
  AreaShape_Eccentricity = rnorm(100)
)
variables <- c("AreaShape_Area", "AreaShape_Eccentricity")
sample <- population %>% dplyr::filter(Metadata_type == "control")
population_marked <-
```

```
    cytominer::mark_outlier_rows(
      population,
      variables,
      sample,
      method = "svd+iqr"
    )
  population_marked %>%
    dplyr::group_by(is_outlier) %>%
    dplyr::sample_n(3)
```

`normalize` *Normalize observation variables.*

Description

`normalize` normalizes observation variables based on the specified normalization method.

Usage

```
normalize(
  population,
  variables,
  strata,
  sample,
  operation = "standardize",
  ...
)
```

Arguments

`population`	tbl with grouping (metadata) and observation variables.
`variables`	character vector specifying observation variables.
`strata`	character vector specifying grouping variables for grouping prior to normalization.
`sample`	tbl containing sample that is used by normalization methods to estimate parameters. `sample` has same structure as `population`. Typically, `sample` corresponds to controls in the experiment.
`operation`	optional character string specifying method for normalization. This must be one of the strings `"standardize"` (default), `"robustize"`.
`...`	arguments passed to normalization operation

Value

normalized data 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)
)
variables <- c("AreaShape_Area")
strata <- c("Metadata_batch")
sample <- population %>% dplyr::filter(Metadata_group == "control")
cytominer::normalize(population, variables, strata, sample, operation = "standardize")
```

`replicate_correlation` *Measure replicate correlation of variables.*

Description

'replicate_correlation' measures replicate correlation of variables.

Usage

```
replicate_correlation(
  sample,
  variables,
  strata,
  replicates,
  replicate_by = NULL,
  split_by = NULL,
  cores = NULL
)
```

Arguments

`sample`	tbl containing sample used to estimate parameters.
`variables`	character vector specifying observation variables.
`strata`	character vector specifying grouping variables for grouping prior to normalization.
`replicates`	number of replicates.
`replicate_by`	optional character string specifying column containing the replicate id.
`split_by`	optional character string specifying column by which to split the sample into batches; replicate correlations will be calculate per batch.
`cores`	optional integer specifying number of CPU cores used for parallel computing using `doParallel`.

Value

data frame of variable quality measurements

Examples

```
set.seed(123)
x1 <- rnorm(10)
x2 <- x1 + rnorm(10) / 100
y1 <- rnorm(10)
y2 <- y1 + rnorm(10) / 10
z1 <- rnorm(10)
z2 <- z1 + rnorm(10) / 1

batch <- rep(rep(1:2, each = 5), 2)

treatment <- rep(1:10, 2)

replicate_id <- rep(1:2, each = 10)

sample <-
  tibble::tibble(
    x = c(x1, x2), y = c(y1, y2), z = c(z1, z2),
    Metadata_treatment = treatment,
    Metadata_replicate_id = replicate_id,
    Metadata_batch = batch
  )

head(sample)

# `replicate_correlation`` returns the median, min, and max
# replicate correlation (across batches) per variable
replicate_correlation(
  sample = sample,
  variables = c("x", "y", "z"),
  strata = c("Metadata_treatment"),
  replicates = 2,
  split_by = "Metadata_batch",
  replicate_by = "Metadata_replicate_id",
  cores = 1
)
```

`sparse_random_projection`

Reduce the dimensionality of a population using sparse random projection.

Description

`sparse_random_projection` reduces the dimensionality of a population by projecting the original data with a sparse random matrix. Generally more efficient and faster to compute than a Gaussian random projection matrix, while providing similar embedding quality.

Usage

```
sparse_random_projection(population, variables, n_components)
```

Arguments

`population`	tbl with grouping (metadata) and observation variables.
`variables`	character vector specifying observation variables.
`n_components`	size of the projected feature space.

Value

Dimensionality reduced `population`.

Examples

```
population <- tibble::tibble(
  Metadata_Well = c("A01", "A02", "B01", "B02"),
  AreaShape_Area_DNA = c(10, 12, 7, 7),
  AreaShape_Length_DNA = c(2, 3, 1, 5),
  Intensity_DNA = c(8, 20, 12, 32),
  Texture_DNA = c(5, 2, 43, 13)
)
variables <- c("AreaShape_Area_DNA", "AreaShape_Length_DNA", "Intensity_DNA", "Texture_DNA")
sparse_random_projection(population, variables, 2)
```

`spherize` *Spherize data.*

Description

`spherize` transforms specified observation variables by estimating a sphering transformation on a sample and applying it to the population.

Usage

```
spherize(population, variables, sample, regularization_param = 1)
```

Arguments

`population`	tbl with grouping (metadata) and observation variables.
`variables`	character vector specifying observation variables.
`sample`	tbl containing sample that is used by the method to estimate sphering parameters. `sample` has same structure as `population`. Typically, `sample` corresponds to controls in the experiment.
`regularization_param`	optional parameter used in sphering to offset eigenvalues to avoid division by zero.

Value

transformed data of the same class as `population`.

Examples

```
population <- tibble::tibble(
  Metadata_Well = c("A01", "A02", "B01", "B02"),
  Intensity_DNA = c(8, 20, 12, 32),
  Texture_DNA = c(5, 2, 43, 13)
)
variables <- c("Intensity_DNA", "Texture_DNA")
spherize(population, variables, population, 0.01)
```

`stratify` *Stratify operations.*

Description

`stratify` stratifies operations.

Usage

```
stratify(population, sample, reducer, strata, ...)
```

Arguments

`population`	tbl with grouping (metadata) and observation variables.
`sample`	tbl with the same structure as `population`. This is typically used by operations to estimate parameters.
`reducer`	operation that is to applied in a stratified manner.
`strata`	optional character vector specifying grouping variables for stratification.
`...`	arguments passed to operation.

Value

`population` with potentially extra columns.

Examples

```
suppressMessages(suppressWarnings(library(magrittr)))
population <- tibble::tibble(
  Metadata_group = sample(c("a", "b"), 100, replace = TRUE),
  Metadata_type = sample(c("control", "trt"), 100, replace = TRUE),
  AreaShape_Area = c(rnorm(98), 20, 30),
  AreaShape_Eccentricity = rnorm(100)
)
variables <- c("AreaShape_Area", "AreaShape_Eccentricity")
strata <- c("Metadata_group")
sample <- population %>% dplyr::filter(Metadata_type == "control")
population_marked <-
  cytominer::stratify(
    reducer = cytominer::mark_outlier_rows,
    method = "svd+iqr",
    population = population,
    variables = variables,
    sample = sample,
    strata = strata
  )
population_marked %>%
  dplyr::group_by(is_outlier) %>%
  dplyr::sample_n(3)
```

`svd_entropy` *Feature importance based on data entropy.*

Description

`svd_entropy` measures the contribution of each feature in decreasing the data entropy.

Usage

```
svd_entropy(sample, variables, cores = NULL)
```

Arguments

`sample`	tbl containing sample used to estimate parameters.
`variables`	character vector specifying observation variables.
`cores`	optional integer specifying number of CPU cores used for parallel computing using `doParallel`.

Value

data frame specifying the contribution of each feature in decreasing the data entropy. Higher values indicate more information.

Examples

```
sample <- tibble::tibble(
  AreaShape_MinorAxisLength = c(10, 12, 15, 16, 8, 8, 7, 7, 13, 18),
  AreaShape_MajorAxisLength = c(35, 18, 22, 16, 9, 20, 11, 15, 18, 42),
  AreaShape_Area = c(245, 151, 231, 179, 50, 112, 53, 73, 164, 529)
)
variables <- c("AreaShape_MinorAxisLength", "AreaShape_MajorAxisLength", "AreaShape_Area")
svd_entropy(sample, variables, cores = 1)
```

`transform` *Transform observation variables.*

Description

`transform` transforms observation variables based on the specified transformation method.

Usage

```
transform(population, variables, operation = "generalized_log", ...)
```

Arguments

`population`	tbl with grouping (metadata) and observation variables.
`variables`	character vector specifying observation variables.
`operation`	optional character string specifying method for transform. This must be one of the strings `"generalized_log"` (default), `"spherize"`, or `"sparse_random_projection"`.
`...`	arguments passed to transformation operation.

Value

transformed data of the same class as `population`.

Examples

```
population <- tibble::tibble(
  Metadata_Well = c("A01", "A02", "B01", "B02"),
  Intensity_DNA = c(8, 20, 12, 32),
  Intensity_RNA = c(1, 12, -1, 4),
  Intensity_AGP = c(-2, 5, -5, -2),
  Intensity_Mito = c(-1, 15, 5, 22),
  Intensity_ER = c(-12, 15, -25, 24)
)
variables <- c("Intensity_DNA", "Intensity_RNA", "Intensity_AGP", "Intensity_ER")
transform(population, variables, operation = "generalized_log")
transform(population, variables, sample = population, operation = "husk", remove_outliers = FALSE)
transform(population, variables, sample = population, operation = "spherize")
transform(population, variables, n_components = 2, operation = "sparse_random_projection")
```

variable_importance *Measure variable importance.*

Description

`variable_importance` measures importance of variables based on specified methods.

Usage

```
variable_importance(
  sample,
  variables,
  operation = "replicate_correlation",
  ...
)
```

Arguments

`sample`	tbl containing sample used to estimate parameters.
`variables`	character vector specifying observation variables.
`operation`	optional character string specifying method for computing variable importance. This must be one of the strings `"replicate_correlation"` (default) or `"svd_entropy"`. is implemented.
`...`	arguments passed to variable importance operation.

Value

data frame containing variable importance measures.

Examples

```
set.seed(123)
x1 <- rnorm(10)
x2 <- x1 + rnorm(10) / 100
y1 <- rnorm(10)
y2 <- y1 + rnorm(10) / 10
z1 <- rnorm(10)
z2 <- z1 + rnorm(10) / 1

batch <- rep(rep(1:2, each = 5), 2)

treatment <- rep(1:10, 2)

replicate_id <- rep(1:2, each = 10)

sample <-
  tibble::tibble(
    x = c(x1, x2), y = c(y1, y2), z = c(z1, z2),
    Metadata_treatment = treatment,
    Metadata_replicate_id = replicate_id,
    Metadata_batch = batch
  )

head(sample)

# `replicate_correlation`` returns the median, min, and max
# replicate correlation (across batches) per variable
variable_importance(
  sample = sample,
  variables = c("x", "y", "z"),
  operation = "replicate_correlation",
  strata = c("Metadata_treatment"),
  replicates = 2,
  split_by = "Metadata_batch",
  replicate_by = "Metadata_replicate_id",
  cores = 1
)

# `svd_entropy`` measures the contribution of each variable in decreasing
# the data entropy.

variable_importance(
  sample = sample,
  variables = c("x", "y", "z"),
  operation = "svd_entropy",
  cores = 1
)
```

`variable_select` *Select observation variables.*

Description

`variable_select` selects observation variables based on the specified variable selection method.

Usage

```
variable_select(
  population,
  variables,
  sample = NULL,
  operation = "variance_threshold",
  ...
)
```

Arguments

population	tbl with grouping (metadata) and observation variables.
variables	character vector specifying observation variables.
sample	tbl containing sample that is used by some variable selection methods. sample has same structure as population.
operation	optional character string specifying method for variable selection. This must be one of the strings "variance_threshold", "correlation_threshold", "drop_na_columns".
...	arguments passed to selection operation.

Value

variable-selected data of the same class as population.

Examples

```
# In this example, we use `correlation_threshold` as the operation for
# variable selection.

suppressMessages(suppressWarnings(library(magrittr)))
population <- tibble::tibble(
  x = rnorm(100),
  y = rnorm(100) / 1000
)

population %<>% dplyr::mutate(z = x + rnorm(100) / 10)

sample <- population %>% dplyr::slice(1:30)

variables <- c("x", "y", "z")

operation <- "correlation_threshold"

cor(sample)

# `x` and `z` are highly correlated; one of them will be removed

head(population)

futile.logger::flog.threshold(futile.logger::ERROR)

variable_select(population, variables, sample, operation) %>% head()
```

---

variance_threshold *Remove variables with near-zero variance.*

Description

variance_threshold returns list of variables that have near-zero variance.

Usage

```
variance_threshold(variables, sample)
```

Arguments

`variables`	character vector specifying observation variables.
`sample`	tbl containing sample used to estimate parameters.

Details

`variance_threshold` is a reimplementation of `caret::nearZeroVar`, using the default values for `freqCut` and `uniqueCut`.

Value

character vector specifying observation variables to be excluded.

Examples

```
sample <- tibble::tibble(
  AreaShape_Area = c(10, 12, 15, 16, 8, 8, 7, 7, 13, 18),
  AreaShape_Euler = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
)
variables <- c("AreaShape_Area", "AreaShape_Euler")
variance_threshold(variables, sample)
```

Index