Package ‘cytominer’

May 9, 2020

Encoding UTF-8
Type Package
Title Methods for Image-Based Cell Profiling
Version 0.2.2
Description Typical morphological profiling datasets have millions of cells and hundreds of features per cell. When working with this data, you must clean the data, normalize the features to make them comparable across experiments, transform the features, select features based on their quality, and aggregate the single-cell data, if needed. ‘cytominer’ makes these steps fast and easy. Methods used in practice in the field are discussed in Caicedo (2017) <doi:10.1038/nmeth.4397>. An overview of the field is presented in Caicedo (2016) <doi:10.1016/j.copbio.2016.04.003>.

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)
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 7.1.0

NeedsCompilation no

Author Tim Becker [aut], Allen Goodman [aut], Claire McQuin [aut], Mohammad Rohban [aut], Shantanu Singh [aut, cre]
R topics documented:

aggregate ................................................................. 2
correlation_threshold .................................................. 3
count_na_rows ............................................................. 4
covariance ................................................................. 5
drop_na_columns .......................................................... 6
drop_na_rows ............................................................... 7
extract_subpopulations .................................................. 7
generalized_log .......................................................... 8
generate_component_matrix .......................................... 9
normalize ................................................................. 10
replicate_correlation ................................................... 11
sparse_random_projection .............................................. 12
svd_entropy ............................................................... 13
transform ................................................................. 14
variable_importance .................................................... 15
variable_select .......................................................... 16
variance_threshold ...................................................... 17
whiten ................................................................. 18

Index 19

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,
    ...)


correlation_threshold

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

```r
population <- tibble::tibble(
  Metadata_group = c("control", "control", "control", "control",
                     "experiment", "experiment", "experiment"),
  Metadata_batch = c("a", "a", "b", "b", "a", "b", "b"),
  AreaShape_Area = c(10, 12, 15, 16, 8, 8, 7)
)
variables <- c("AreaShape_Area")
strata <- c("Metadata_group", "Metadata_batch")
aggregate(population, variables, strata, operation = "mean")
```

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")
count_na_rows

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 NAs per variable.

Description
count_na_rows counts the number of NAs per variable.

Usage
count_na_rows(population, variables)
covariance

Arguments

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

Value

data frame with frequency of NAs per variable.

Examples

```r
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

```r
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 NAs.

Usage

```r
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

```r
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**

```r
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**

```r
population <- tibble::tibble(
  Metadata_group = c(
    "control", "control", "control", "control",
    "experiment", "experiment", "experiment", "experiment"
  ),
  Metadata_batch = c("a", "a", "b", "b", "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**

```r
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**

```r
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
)
```

**generalized_log**

*Generalized log transform data.*

**Description**

`generalized_log` transforms specified observation variables using \( x = \log((x+\sqrt{x^2+\text{offset}^2})/2) \).

**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.
generate_component_matrix

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
generate_component_matrix(500, 100, 0.3)
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".
...

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")
replicate_correlation

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 calculated per batch.
cores       optional integer specifying number of CPU cores used for parallel computing using doParallel.

Value

data frame of variable quality measurements
Examples

```r
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.
svd_entropy

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)

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(variables, sample, cores = NULL)

Arguments

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

Examples

```r
table <- 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(variables, sample, cores = 1)
```

---

**transform**

Transform observation variables.

**Description**

transform transforms observation variables based on the specified transformation method.

**Usage**

```r
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), "whiten".
- `...` arguments passed to transformation operation.

**Value**

transformed data of the same class as population.

**Examples**

```r
population <- tibble::tibble(
  Metadata_Well = c("A01", "A02", "B01", "B02"),
  Intensity_DNA = c(8, 20, 12, 32)
)
variables <- c("Intensity_DNA")
transform(population, variables, operation = "generalized_log")
```
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.
                      Currently, only "replicate_correlation" (default) 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,
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

```r
# 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**

```r
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
```r
table <- 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, table)
```

whiten
Whiten data.

Description
whiten transforms specified observation variables by estimating a whitening transformation on a
sample and applying it to the population.

Usage
```r
whiten(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 whitening para-
  meters. sample has same structure as population. Typically, sample corresponds
to controls in the experiment.
- `regularization_param`: optional parameter used in whitening to offset eigenvalues to avoid division by
  zero.

Value
transformed data of the same class as population.

Examples
```r
table <- 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")
whiten(table, variables, population, 0.01)
```
Index

aggregate, 2

correlation_threshold, 3
count_na_rows, 4
covariance, 5

drop_na_columns, 6
drop_na_rows, 7

extract_subpopulations, 7
generalized_log, 8
generate_component_matrix, 9

normalize, 10

replicate_correlation, 11

sparse_random_projection, 12
svd_entropy, 13

transform, 14

variable_importance, 15
variable_select, 16
variance_threshold, 17

whiten, 18