Package ‘partition’

October 14, 2022

Type Package

Title Agglomerative Partitioning Framework for Dimension Reduction

Version 0.1.4

Maintainer Malcolm Barrett <malcolmbarrett@gmail.com>

Description A fast and flexible framework for agglomerative partitioning. 'partition' uses an approach called Direct-Measure-Reduce to create new variables that maintain the user-specified minimum level of information. Each reduced variable is also interpretable: the original variables map to one and only one variable in the reduced data set. 'partition' is flexible, as well: how variables are selected to reduce, how information loss is measured, and the way data is reduced can all be customized. 'partition' is based on the Partition framework discussed in Millstein et al. (2020) <doi:10.1093/bioinformatics/btz661>.

License MIT + file LICENSE


BugReports https://github.com/USCbiostats/partition/issues

Depends R (>= 3.3.0)

Imports crayon, dplyr (>= 0.8.0), forcats, ggplot2 (>= 3.3.0), infotheo, magrittr, MASS, pillar, purrr, Rcpp, rlang, stringr, tibble, tidyr (>= 1.0.0)

Suggests covr, knitr, rmarkdown, spelling, testthat (>= 3.0.0), ggcorrplot

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.1.2
Config/testthat/edition  3
Config/testthat/parallel  true

NeedsCompilation  yes

Author  Joshua Millstein [aut],
         Malcolm Barrett [aut, cre] (<https://orcid.org/0000-0003-0299-5825>)

Repository  CRAN
Date/Publication  2021-10-05 04:20:02 UTC

R topics documented:

as_director .................................................. 3
as_measure .................................................... 4
as_partitioner ................................................ 5
as_partition_step .......................................... 6
as_reducer .................................................... 7
baxter_data ................................................... 8
corr ............................................................ 9
direct_distance ............................................ 9
direct_k_cluster .......................................... 10
filter_reduced ................................................ 11
icc .............................................................. 12
is_partition .................................................. 13
is_partitioner ................................................ 13
is_partition_step ......................................... 14
mapping_key ................................................ 14
map_partition ................................................ 15
measure_icc .................................................. 16
measure_min_icc ........................................... 17
measure_min_r2 ............................................. 17
measure_std_mutualinfo .................................. 18
measure_variance_explained ............................. 19
mutual_information ......................................... 19
partition ...................................................... 20
partition_scores ......................................... 22
part_icc ...................................................... 23
part_kmeans .................................................. 24
part_minr2 ................................................... 25
part_pc1 ........................................................ 26
part_stdmi .................................................... 27
permute_df ................................................... 28
plot_area_clusters ......................................... 28
plot_permutation .......................................... 30
reduce_cluster ............................................. 31
reduce_first_component .................................. 32
reduce_kmeans .............................................. 32
reduce_scaled_mean ....................................... 33
as_director

as_director

Description

Directors are functions that tell the partition algorithm what to try to reduce. as_director() is a helper function to create new directors to be used in partitioners. partitioners can be created with as_partitioner().

Usage

as_director(.pairs, .target, ...)

Arguments

.pairs a function that returns a matrix of targets (e.g. a distance matrix of variables)
.target a function that returns a vector of targets (e.g. the minimum pair)
... Extra arguments passed to .f.

Value

a function to use in as_partitioner()

See Also

Other directors: direct_distance(), direct_k_cluster()

Examples

# use euclidean distance to calculate distances
euc_dist <- function(.data) as.matrix(dist(t(.data)))

# find the pair with the minimum distance
min_dist <- function(.x) {
  indices <- arrayInd(which.min(.x), dim(as.matrix(.x)))

  # get variable names with minimum distance
  c(
    colnames(.x)[indices[1]],
    colnames(.x)[indices[2]]
  )
}


as_director(euc_dist, min_dist)

---

**as_measure**  
*Create a custom metric*

**Description**

Metrics are functions that tell how much information would be lost for a given reduction in the data. `as_measure()` is a helper function to create new metrics to be used in partitioners. Partitioners can be created with `as_partitioner()`.

**Usage**

```r
as_measure(.f, ...)
```

**Arguments**

- `.f`  
a function that returns either a numeric vector or a `data.frame`

- `...`  
Extra arguments passed to `.f`.

**Value**

a function to use in `as_partitioner()`

**See Also**

Other metrics: `measure_icc()`, `measure_min_icc()`, `measure_min_r2()`, `measure_std_mutualinfo()`, `measure_variance_explained()`

**Examples**

```r
inter_item_reliability <- function(.data) {
  corr(.data) %>%
  colMeans(na.rm = TRUE) %>%
  mean()
}

measure_iir <- as_measure(inter_item_reliability)
measure_iir
```
as_partitioner  Create a partitioner

Description

Partitioners are functions that tell the partition algorithm 1) what to try to reduce 2) how to measure how much information is lost from the reduction and 3) how to reduce the data. In partition, functions that handle 1) are called directors, functions that handle 2) are called metrics, and functions that handle 3) are called reducers. partition has a number of pre-specified partitioners for agglomerative data reduction. Custom partitioners can be created with as_partitioner().

Pass partitioner objects to the partitioner argument of partition().

Usage

as_partitioner(direct, measure, reduce)

Arguments

direct a function that directs, possibly created by as_director()
measure a function that measures, possibly created by as_measure()
reduce a function that reduces, possibly created by as_reducer()

Value

a partitioner

See Also

Other partitioners: part_icc(), part_kmeans(), part_minr2(), part_pcl(), part_stdmi(), replace_partitioner()

Examples

as_partitioner(
  direct = direct_distance_pearson,
  measure = measure_icc,
  reduce = reduce_scaled_mean
)
as_partition_step

Create a partition object from a data frame

Description

as_partition_step() creates a partition_step object. partition_steps are used while iterating through the partition algorithm: it stores necessary information about how to proceed in the partitioning, such as the information threshold. as_partition_step() is primarily called internally by partition() but can be helpful while developing partitioners.

Usage

as_partition_step(
  .x,
  threshold = NA,
  reduced_data = NA,
  target = NA,
  metric = NA,
  tolerance = 0.01,
  var_prefix = NA,
  partitioner = NA,
  ...
)

Arguments

.x a data.frame or partition_step object
threshold The minimum information loss allowable
reduced_data A data set with reduced variables
target A character or integer vector: the variables to reduce
metric A measure of information
tolerance A tolerance around the threshold to accept a reduction
var_prefix Variable name for reduced variables
partitioner A partitioner, a part_*() function or one created with as_partitioner().
... Other objects to store during the partition step

Value

a partition_step object

Examples

.df <- data.frame(x = rnorm(100), y = rnorm(100))
as_partition_step(.df, threshold = .6)
Create a custom reducer

Description

Reducers are functions that tell the partition algorithm how to reduce the data. `as_reducer()` is a helper function to create new reducers to be used in partitioners. Partitioners can be created with `as_partitioner()`.

Usage

`as_reducer(.f, ..., returns_vector = TRUE, first_match = NULL)`

Arguments

- `.f` a function that returns either a numeric vector or a data.frame
- `...` Extra arguments passed to `.f`
- `returns_vector` logical. Does `.f` return a vector? TRUE by default. If FALSE, assumes that `.f` returns a data.frame.
- `first_match` logical. Should the partition algorithm stop when it finds a reduction that is equal to the threshold? Default is TRUE for reducers that return a data.frame and FALSE for reducers that return a vector

Value

A function to use in `as_partitioner()`

See Also

Other reducers: `reduce_first_component()`, `reduce_kmeans()`, `reduce_scaled_mean()`

Examples

```r
reduce_row_means <- as_reducer(rowMeans)
reduce_row_means
```
**baxter_data**

---

**Microbiome data**

---

**Description**

Clinical and microbiome data derived from "Microbiota-based model improves the sensitivity of fecal immunochemical test for detecting colonic lesions" by Baxter et al. (2016). These data represent a subset of 172 health participants. **baxter_clinical** contains 8 clinical variables for each of the participants: sample_name, id, age, bmi, gender, height, total_reads, and disease_state (all H for healthy). **baxter_otu** has 1,234 columns, where each column represents an Operational Taxonomic Unit (OTU). OTUs are species-like relationships among bacteria determined by analyzing their RNA. The cells are logged counts for how often the OTU was detected in a participant’s stool sample. Each column name is a shorthand name, e.g. otu1; you can find the true name of the OTU mapped in **baxter_data_dictionary**. **baxter_family** and **baxter_genus** are also logged counts but instead group OTUs at the family and genus level, respectively, a common approach to reducing microbiome data. Likewise, the column names are shorthands, which you can find mapped in **baxter_data_dictionary**.

**Usage**

- **baxter_clinical**
- **baxter_otu**
- **baxter_family**
- **baxter_genus**
- **baxter_data_dictionary**

**Format**

- **5 data frames**
  - An object of class tbl_df (inherits from tbl.data.frame) with 172 rows and 1234 columns.
  - An object of class tbl_df (inherits from tbl.data.frame) with 172 rows and 35 columns.
  - An object of class tbl_df (inherits from tbl.data.frame) with 172 rows and 82 columns.
  - An object of class tbl_df (inherits from tbl.data.frame) with 1351 rows and 3 columns.

**Source**

# corr

**Efficiently fit correlation coefficient for matrix or two vectors**

## Description

Efficiently fit correlation coefficient for matrix or two vectors

## Usage

```r
corr(x, y = NULL, spearman = FALSE)
```

## Arguments

- `x`: a matrix or vector
- `spearman`: Logical. Use Spearman’s correlation?

## Value

a numeric vector, the correlation coefficient

## Examples

```r
library(dplyr)
# fit for entire data set
iris %>%
  select_if(is.numeric) %>%
  corr()

# just fit for two vectors
corr(iris$Sepal.Length, iris$Sepal.Width)
```

## direct_distance

**Target based on minimum distance matrix**

## Description

Directors are functions that tell the partition algorithm what to try to reduce. `as_director()` is a helper function to create new directors to be used in partitioners. Partitioners can be created with `as_partitioner()`.

`direct_distance()` fits a distance matrix using either Pearson’s or Spearman’s correlation and finds the pair with the smallest distance to target. If the distance matrix already exists, `direct_distance()` only fits the distances for any new reduced variables. `direct_distance_pearson()` and `direct_distance_spearman()` are convenience functions that directly call the type of distance matrix.
direct_k_cluster

Usage

direct_distance(.partition_step, spearman = FALSE)
direct_distance_pearson(.partition_step)
direct_distance_spearman(.partition_step)

Arguments

.partition_step
  a partition_step object

spearman
  Logical. Use Spearman’s correlation?

Value

  a partition_step object

See Also

  Other directors: as_director(), direct_k_cluster()

---

direct_k_cluster  Target based on K-means clustering

Description

Directors are functions that tell the partition algorithm what to try to reduce. as_director() is a helper function to create new directors to be used in partitioners. partitioners can be created with as_partitioner().

direct_k_cluster() assigns each variable to a cluster using K-means. As the partition looks for the best reduction, direct_k_cluster() iterates through values of k to assign clusters. This search is handled by the binary search method by default and thus does not necessarily need to fit every value of k.

Usage

direct_k_cluster(
  .partition_step,
  algorithm = c("armadillo", "Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"),
  search = c("binary", "linear"),
  init_k = NULL,
  seed = 1L
)
**filter_reduced**

Arguments

- **.partition_step**
  
  a partition_step object

- **algorithm**
  
  The K-Means algorithm to use. The default is a fast version of the LLoyd algorithm written in armadillo. The rest are options in `kmeans()`. In general, armadillo is fastest, but the other algorithms can be faster in high dimensions.

- **search**
  
  The search method. Binary search is generally more efficient but linear search can be faster in very low dimensions.

- **init_k**
  
  The initial k to test. If NULL, then the initial k is the threshold times the number of variables.

- **seed**
  
  The seed to set for reproducibility

Value

- a partition_step object

See Also

Other directors: `as_director()`, `direct_distance()`

---

**filter_reduced** *Filter the reduced mappings*

**Description**

`filter_reduced()` and `unnest_reduced()` are convenience functions to quickly retrieve the mappings for only the reduced variables. `filter_reduced()` returns a nested tibble while `unnest_reduced()` unnests it.

**Usage**

```r
filter_reduced(.partition)

unnest_reduced(.partition)
```

**Arguments**

- **.partition** a partition object

**Value**

- a tibble with mapping key
Examples

```r
set.seed(123)
df <- simulate_block_data(c(3, 4, 5), lower_corr = .4, upper_corr = .6, n = 100)
# fit partition
prt <- partition(df, threshold = .6)

# A tibble: 3 x 4
filter_reduced(prt)

# A tibble: 9 x 4
unnest_reduced(prt)
```

---

**icc**  
*Calculate the intraclass correlation coefficient*

**Description**

`icc()` efficiently calculates the ICC for a numeric data set.

**Usage**

```r
icc(.x, method = c("r", "c"))
```

**Arguments**

- `.x` a data set
- `method` The method source: both the pure R and C++ versions are efficient

**Value**

a numeric vector of length 1

**Examples**

```r
library(dplyr)
iris %>%
  select_if(is.numeric) %>%
  icc()
```
is_partition

Description
Is this object a partition?

Usage
is_partition(x)

Arguments
x an object to be tested

Value
logical: TRUE or FALSE

is_partitioner

Description
Is this object a partitioner?

Usage
is_partitioner(x)

Arguments
x an object to be tested

Value
logical: TRUE or FALSE
### is_partition_step

**Is this object a partition_step?**

**Description**

Is this object a partition_step?

**Usage**

```r
is_partition_step(x)
```

**Arguments**

- `x`: an object to be tested

**Value**

logical: TRUE or FALSE

### mapping_key

**Return partition mapping key**

**Description**

mapping_key() returns a data frame with each reduced variable and its mapping and information loss; the mapping and indices are represented as list-cols (so there is one row per variable in the reduced data set). unnest_mappings() unnestes the list columns to return a tidy data frame. mapping_groups() returns a list of mappings (either the variable names or their column position).

**Usage**

```r
mapping_key(.partition)

unnest_mappings(.partition)

mapping_groups(.partition, indices = FALSE)
```

**Arguments**

- `.partition`: a partition object
- `.indices`: logical. Return just the indices instead of the names? Default is FALSE.

**Value**

a tibble
Examples

```r
set.seed(123)
df <- simulate_block_data(c(3, 4, 5), lower_corr = .4, upper_corr = .6, n = 100)
# fit partition
prt <- partition(df, threshold = .6)
# tibble: 6 x 4
mapping_key(prt)
# tibble: 12 x 4
unnest_mappings(prt)
# list: length 6
mapping_groups(prt)
```

map_partition

Map a partition across a range of minimum information

Description

map_partition() fits partition() across a range of minimum information values, specified in the information argument. The output is a tibble with a row for each value of information, a summary of the partition, and a list-col containing the partition object.

Usage

```r
map_partition(
  .data,
  partitioner = part_icc(),
  ...,
  information = seq(0.1, 0.5, by = 0.1)
)
```

Arguments

- `.data` a data set to partition
- `partitioner` the partitioner to use. The default is `part_icc()`.
- `...` arguments passed to `partition()`
- `information` a vector of minimum information to fit in `partition()`

Value

a tibble
Examples

```r
set.seed(123)
df <- simulate_block_data(c(3, 4, 5), lower_corr = .4, upper_corr = .6, n = 100)

map_partition(df, partitioner = part_pc1())
```

---

**measure_icc**

*Measure the information loss of reduction using intraclass correlation coefficient*

**Description**

Metrics are functions that tell how much information would be lost for a given reduction in the data. `as_measure()` is a helper function to create new metrics to be used in partitioners. Partitioners can be created with `as_partitioner()`.

`measure_icc()` assesses information loss by calculating the intraclass correlation coefficient for the target variables.

**Usage**

```
measure_icc(.partition_step)
```

**Arguments**

- `.partition_step`

  a partition_step object

**Value**

a partition_step object

**See Also**

Other metrics: `as_measure()`, `measure_min_icc()`, `measure_min_r2()`, `measure_std_mutualinfo()`, `measure_variance_explained()`
measure_min_icc

Measure the information loss of reduction using the minimum intra-class correlation coefficient

Description

Metrics are functions that tell how much information would be lost for a given reduction in the data. reduce. `as_measure()` is a helper function to create new metrics to be used in partitioners. Partitioners can be created with `as_partitioner()`.

`measure_min_icc()` assesses information loss by calculating the intraclass correlation coefficient for each set of the target variables and finding their minimum.

Usage

`measure_min_icc(.partition_step, search_method = c("binary", "linear"))`

Arguments

- `.partition_step` a `partition_step` object
- `search_method` The search method. Binary search is generally more efficient but linear search can be faster in very low dimensions.

Value

a `partition_step` object

See Also

Other metrics: `as_measure()`, `measure_icc()`, `measure_min_r2()`, `measure_std_mutualinfo()`, `measure_variance_explained()`

measure_min_r2

Measure the information loss of reduction using minimum R-squared

Description

Metrics are functions that tell how much information would be lost for a given reduction in the data. reduce. `as_measure()` is a helper function to create new metrics to be used in partitioners. Partitioners can be created with `as_partitioner()`.

`measure_min_r2()` assesses information loss by calculating the minimum R-squared for the target variables.

Usage

`measure_min_r2(.partition_step)`
measure_std_mutualinfo

Arguments

.partition_step

a partition_step object

Value

a partition_step object

See Also

Other metrics: as_measure(), measure_icc(), measure_min_icc(), measure_std_mutualinfo(), measure_variance_explained()

measure_std_mutualinfo

Measure the information loss of reduction using standardized mutual information

Description

Metrics are functions that tell how much information would be lost for a given reduction in the data. reduce. as_measure() is a helper function to create new metrics to be used in partitioners. partitioners can be created with as_partitioner().

measure_std_mutualinfo() assesses information loss by calculating the standardized mutual information for the target variables. See mutual_information().

Usage

measure_std_mutualinfo(.partition_step)

Arguments

.partition_step

a partition_step object

Value

a partition_step object

See Also

Other metrics: as_measure(), measure_icc(), measure_min_icc(), measure_min_r2(), measure_variance_explained()
measure_variance_explained

Measure the information loss of reduction using the variance explained

Description

Metrics are functions that tell how much information would be lost for a given reduction in the data. reduce. `as_measure()` is a helper function to create new metrics to be used in partitioners. Partitioners can be created with `as_partitioner()`.

`measure_variance_explained()` assesses information loss by calculating the variance explained by the first component of a principal components analysis.

Usage

`measure_variance_explained(.partition_step)`

Arguments

- `.partition_step`
  - a partition_step object

Value

- a partition_step object

See Also

Other metrics: `as_measure()`, `measure_icc()`, `measure_min_icc()`, `measure_min_r2()`, `measure_std_mutualinfo()`

mutual_information

Calculate the standardized mutual information of a data set

Description

`mutual_information` calculate the standardized mutual information of a data set using the infotheo package.

Usage

`mutual_information(.data)`

Arguments

- `.data`
  - a dataframe of numeric values
**Value**

a list containing the standardized MI and the scaled row means

**Examples**

```r
library(dplyr)
iris %>%
  select_if(is.numeric) %>%
  mutual_information()
```

---

**Description**

`partition()` reduces data while minimizing information loss using an agglomerative partitioning algorithm. The partition algorithm is fast and flexible: at every iteration, `partition()` uses an approach called Direct-Measure-Reduce (see Details) to create new variables that maintain the user-specified minimum level of information. Each reduced variable is also interpretable: the original variables map to one and only one variable in the reduced data set.

**Usage**

```r
partition(
  .data,
  threshold,
  partitioner = part_icc(),
  tolerance = 1e-04,
  niter = NULL,
  x = "reduced_var",
  .sep = "_"
)
```

**Arguments**

- **.data** a data.frame to partition
- **threshold** the minimum proportion of information explained by a reduced variable; threshold sets a boundary for information loss because each reduced variable must explain at least as much as `threshold` as measured by the metric.
- **partitioner** a partitioner. See the `part_*()` functions and `as_partitioner()`.
- **tolerance** a small tolerance within the threshold; if a reduction is within the threshold plus/minus the tolerance, it will reduce.
- **niter** the number of iterations. By default, it is calculated as 20% of the number of variables or 10, whichever is larger.
- **x** the prefix of the new variable names
- **.sep** a character vector that separates `x` from the number (e.g. "reduced_var_1").
**Details**

`partition()` uses an approach called Direct-Measure-Reduce. Directors tell the partition algorithm what to reduce, metrics tell it whether or not there will be enough information left after the reduction, and reducers tell it how to reduce the data. Together these are called a partitioner. The default partitioner for `partition()` is `part_icc()`: it finds pairs of variables to reduce by finding the pair with the minimum distance between them, it measures information loss through ICC, and it reduces data using scaled row means. There are several other partitioners available (`part_*()` functions), and you can create custom partitioners with `as_partitioner()` and `replace_partitioner()`.

**Value**

a partition object

**References**


**See Also**

`part_icc()`, `part_kmeans()`, `part_minr2()`, `part_pc1()`, `part_stdmi()`, `as_partitioner()`, `replace_partitioner()`

**Examples**

```r
set.seed(123)
df <- simulate_block_data(c(3, 4, 5), lower_corr = .4, upper_corr = 6, n = 100)

# don't accept reductions where information < .6
prt <- partition(df, threshold = .6)
prt

# return reduced data
partition_scores(prt)

# access mapping keys
mapping_key(prt)
unnest_mappings(prt)

# use a lower threshold of information loss
partition(df, threshold = .5, partitioner = part_kmeans())

# use a custom partitioner
part_icc_rowmeans <- replace_partitioner(part_icc, reduce = as_reducer(rowMeans))
partition(df, threshold = .6, partitioner = part_icc_rowmeans)
```
partition_scores

Return the reduced data from a partition

Description

The reduced data is stored as reduced_data in the partition object and can thus be returned by sub-setting object$reduced_data. Alternatively, the functions partition_score() and fitted() also return the reduced data.

Usage

partition_scores(object, ...)

## S3 method for class 'partition'
fitted(object, ...)

Arguments

object a partition object

... not currently used (for S3 consistency with fitted())

Value

a tibble containing the reduced data for the partition

Examples

set.seed(123)
df <- simulate_block_data(c(3, 4, 5), lower_corr = .4, upper_corr = .6, n = 100)
# fit partition
prt <- partition(df, threshold = .6)

# three ways to retrieve reduced data
partition_scores(prt)
fitted(prt)
prt$reduced_data
**part_icc**

Partitioner: distance, ICC, scaled means

**Description**
Partitioners are functions that tell the partition algorithm 1) what to try to reduce 2) how to measure how much information is lost from the reduction and 3) how to reduce the data. In partition, functions that handle 1) are called directors, functions that handle 2) are called metrics, and functions that handle 3) are called reducers. partition has a number of pre-specified partitioners for agglomerative data reduction. Custom partitioners can be created with `as_partitioner()`.

Pass partitioner objects to the partitioner argument of `partition()`.

`part_icc()` uses the following direct-measure-reduce approach:

- **direct**: `direct_distance()`, Minimum Distance
- **measure**: `measure_icc()`, Intraclass Correlation
- **reduce**: `reduce_scaled_mean()`, Scaled Row Means

**Usage**

```r
part_icc(spearman = FALSE)
```

**Arguments**

- `spearman` logical. Use Spearman’s correlation for distance matrix?

**Value**

a partitioner

**See Also**

Other partitioners: `as_partitioner()`, `part_kmeans()`, `part_minr2()`, `part_pc1()`, `part_stdmi()`, `replace_partitioner()`

**Examples**

```r
set.seed(123)
df <- simulate_block_data(c(3, 4, 5), lower_corr = .4, upper_corr = .6, n = 100)

# fit partition using part_icc()
partition(df, threshold = .6, partitioner = part_icc())
```
Description

Partitioners are functions that tell the partition algorithm 1) what to try to reduce 2) how to measure how much information is lost from the reduction and 3) how to reduce the data. In partition, functions that handle 1) are called directors, functions that handle 2) are called metrics, and functions that handle 3) are called reducers. partition has a number of pre-specified partitioners for agglomerative data reduction. Custom partitioners can be created with `as_partitioner()`.

Pass partitioner objects to the partitioner argument of `partition()`.

`part_kmeans()` uses the following direct-measure-reduce approach:

- **direct**: `direct_k_cluster()`, K-Means Clusters
- **measure**: `measure_min_icc()`, Minimum Intraclass Correlation
- **reduce**: `reduce_kmeans()`, Scaled Row Means

Usage

```r
part_kmeans(
  algorithm = c("armadillo", "Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"),
  search = c("binary", "linear"),
  init_k = NULL,
  n_hits = 4
)
```

Arguments

- **algorithm**
  The K-Means algorithm to use. The default is a fast version of the LLoyd algorithm written in armadillo. The rest are options in `kmeans()`. In general, armadillo is fastest, but the other algorithms can be faster in high dimensions.

- **search**
  The search method. Binary search is generally more efficient but linear search can be faster in very low dimensions.

- **init_k**
  The initial k to test. If NULL, then the initial k is the threshold times the number of variables.

- **n_hits**
  In linear search method, the number of iterations that should be under the threshold before reducing; useful for preventing false positives.

Value

A partitioner

See Also

Other partitioners: `as_partitioner()`, `part_icc()`, `part_minr2()`, `part_pc1()`, `part_stdmi()`, `replace_partitioner()`
**Examples**

```r
set.seed(123)
df <- simulate_block_data(c(3, 4, 5), lower_corr = .4, upper_corr = .6, n = 100)

# fit partition using part_kmeans()
partition(df, threshold = .6, partitioner = part_kmeans())
```

---

**part_minr2**  
**Partitioner: distance, minimum R-squared, scaled means**

**Description**

Partitioners are functions that tell the partition algorithm 1) what to try to reduce 2) how to measure how much information is lost from the reduction and 3) how to reduce the data. In partition, functions that handle 1) are called directors, functions that handle 2) are called metrics, and functions that handle 3) are called reducers. `partition` has a number of pre-specified partitioners for agglomerative data reduction. Custom partitioners can be created with `as_partitioner()`.

Pass `partitioner` objects to the `partitioner` argument of `partition()`.

`part_minr2()` uses the following direct-measure-reduce approach:

- **direct**: `direct_distance()`, Minimum Distance
- **measure**: `measure_min_r2()`, Minimum R-Squared
- **reduce**: `reduce_scaled_mean()`, Scaled Row Means

**Usage**

```r
part_minr2(spearman = FALSE)
```

**Arguments**

- `spearman`  
  logical. Use Spearman's correlation for distance matrix?

**Value**

- a `partitioner`

**See Also**

Other partitioners: `as_partitioner()`, `part_icc()`, `part_kmeans()`, `part_pc1()`, `part_stdmi()`, `replace_partitioner()`
Examples

```r
set.seed(123)
df <- simulate_block_data(c(3, 4, 5), lower_corr = .4, upper_corr = .6, n = 100)

# fit partition using part_minr2()
partition(df, threshold = .6, partitioner = part_minr2())
```

---

**part_pc1**

*Partitioner: distance, first principal component, scaled means*

Description

Partitioners are functions that tell the partition algorithm 1) what to try to reduce 2) how to measure how much information is lost from the reduction and 3) how to reduce the data. In partition, functions that handle 1) are called directors, functions that handle 2) are called metrics, and functions that handle 3) are called reducers. `partition` has a number of pre-specified partitioners for agglomerative data reduction. Custom partitioners can be created with `as_partitioner()`.

Pass partitioner objects to the partitioner argument of `partition()`.

`part_pc1()` uses the following direct-measure-reduce approach:

- **direct**: `direct_distance()`, Minimum Distance
- **measure**: `measure_variance_explained()`, Variance Explained (PCA)
- **reduce**: `reduce_first_component()`, First Principal Component

Usage

```r
part_pc1(spearman = FALSE)
```

Arguments

- **spearman**: logical. Use Spearman’s correlation for distance matrix?

Value

- a partitioner

See Also

Other partitioners: `as_partitioner()`, `part_icc()`, `part_kmeans()`, `part_minr2()`, `part_stdmi()`, `replace_partitioner()`
part_stdmi

Examples

```r
set.seed(123)
df <- simulate_block_data(c(3, 4, 5), lower_corr = .4, upper_corr = .6, n = 100)

# fit partition using part_pc1()
partition(df, threshold = .6, partitioner = part_pc1())
```

---

part_stdmi  
**Partitioner: distance, mutual information, scaled means**

Description

Partitioners are functions that tell the partition algorithm 1) what to try to reduce 2) how to measure how much information is lost from the reduction and 3) how to reduce the data. In partition, functions that handle 1) are called directors, functions that handle 2) are called metrics, and functions that handle 3) are called reducers. partition has a number of pre-specified partitioners for agglomerative data reduction. Custom partitioners can be created with `as_partitioner()`.

Pass partitioner objects to the `partitioner` argument of `partition()`.

`part_stdmi()` uses the following direct-measure-reduce approach:

- **direct**: `direct_distance()`, Minimum Distance
- **measure**: `measure_std_mutualinfo()`, Standardized Mutual Information
- **reduce**: `reduce_scaled_mean()`, Scaled Row Means

Usage

```r
part_stdmi(spearman = FALSE)
```

Arguments

- `spearman`  
  logical. Use Spearman’s correlation for distance matrix?

Value

a partitioner

See Also

Other partitioners: `as_partitioner()`, `part_icc()`, `part_kmeans()`, `part_minr2()`, `part_pc1()`, `replace_partitioner()`
Example

```r
set.seed(123)
df <- simulate_block_data(c(3, 4, 5), lower_corr = .4, upper_corr = .6, n = 100)

# fit partition using part_stdmi()
partition(df, threshold = .6, partitioner = part_stdmi())
```

---

### permute_df

#### Permute a data set

**Description**

`permute_df()` permutes a data set: it randomizes the order within each variable, which breaks any association between them. Permutation is useful for testing against null statistics.

**Usage**

```r
permute_df(.data)
```

**Arguments**

- `.data` a data.frame

**Value**

a permuted data.frame

**Examples**

```r
permute_df(iris)
```

---

### plot_area_clusters

#### Plot partitions

**Description**

`plot_stacked_area_clusters()` and `plot_area_clusters()` plot the partition against a permuted partition. `plot_ncluster()` plots the number of variables per cluster. If `.partition` is the result of `map_partition()` or `test_permutation()`, `plot_ncluster()` facets the plot by each partition. `plot_information()` plots a histogram or density plot of the information of each variable in the partition. If `.partition` is the result of `map_partition()` or `test_permutation()`, `plot_information()` plots a scatterplot of the targeted vs. observed information with a 45 degree line indicating perfect alignment.
Usage

plot_area_clusters(
  .data,
  partitioner = part_icc(),
  information = seq(0.1, 0.5, length.out = 25),
  ..., 
  obs_color = "#E69F00",
  perm_color = "#56B4E9"
)

plot_stacked_area_clusters(
  .data,
  partitioner = part_icc(),
  information = seq(0.1, 0.5, length.out = 25),
  ..., 
  stack_colors = c("#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00")
)

plot_ncluster(
  .partition,
  show_n = 100,
  fill = "#0172B1",
  color = NA,
  labeller = "target information:"
)

plot_information(
  .partition,
  fill = "#0172B1",
  color = NA,
  geom = ggplot2::geom_density
)

Arguments

.data a data.frame to partition
partitioner a partitioner. See the part_*() functions and as_partitioner().
information a vector of minimum information to fit in partition()
... arguments passed to partition()
obs_color the color of the observed partition
perm_color the color of the permuted partition
stack_colors the colors of the cluster sizes
.partition either a partition or a tibble, the result of map_partition() or test_permutation()
show_n the number of reduced variables to plot
fill the color of the fill for geom
plot_permutation

the color of the geom
the facet label
the geom to use. The default is geom_density.

Value
a ggplot

Examples

set.seed(123)
df <- simulate_block_data(c(3, 4, 5), lower_corr = .4, upper_corr = .6, n = 100)

  df %>%
    partition(.6, partitioner = part_pc1()) %>%
    plot_ncluster()

Description

plot_permutation() takes the results of test_permutation() and plots the distribution of permuted partitions compared to the observed partition.

Usage

plot_permutation(
  permutations,
  .plot = c("information", "nclusters", "nreduced"),
  labeller = "target information:",
  perm_color = "#56B4EA",
  obs_color = "#CC78A8",
  geom = ggplot2::geom_density
)

Arguments

permutations a tibble, the result of test_permutation()
.plot the variable to plot: observed information, the number of clusters created, or the number of observed variables reduced
labeller the facet label
perm_color the color of the permutation fill
obs_color the color of the observed statistic line
geom the geom to use. The default is geom_density.
reduce_cluster

Value

a ggplot

reduce_cluster       Reduce a target

Description

reduce_cluster() and map_cluster() apply the data reduction to the targets found in the director step. They only do so if the metric is above the threshold, however. reduce_cluster() is for functions that return vectors while map_cluster() is for functions that return data.frames. If you’re using as_reducer(), there’s no need to call these functions directly.

Usage

reduce_cluster(.partition_step, .f, first_match = FALSE)

map_cluster(.partition_step, .f, rewind = FALSE, first_match = FALSE)

Arguments

.partition_step
  a partition_step object

.f
  a function to reduce the data to either a vector or a data.frame

first_match
  logical. Should the partition algorithm stop when it finds a reduction that is equal to the threshold? Default is TRUE for reducers that return a data.frame and FALSE for reducers that return a vector

rewind
  logical. Should the last target be used instead of the current target?

Value

a partition_step object

Examples

reduce_row_means <- function(.partition_step, .data) {
  reduce_cluster(.partition_step, rowMeans)
}

replace_partitioner(
  part_icc,
  reduce = reduce_row_means
)
reduce_first_component

Reduce selected variables to first principal component

Description

Reducers are functions that tell the partition algorithm how to reduce the data. `as_reducer()` is a helper function to create new reducers to be used in partitioners. Partitioners can be created with `as_partitioner()`.

`reduce_first_component()` returns the first component from the principal components analysis of the target variables.

Usage

```r
reduce_first_component(.partition_step)
```

Arguments

- `.partition_step`
  
a `partition_step` object

Value

a `partition_step` object

See Also

Other reducers: `as_reducer()`, `reduce_kmeans()`, `reduce_scaled_mean()`

reduce_kmeans

Reduce selected variables to scaled means

Description

Reducers are functions that tell the partition algorithm how to reduce the data. `as_reducer()` is a helper function to create new reducers to be used in partitioners. Partitioners can be created with `as_partitioner()`.

`reduce_kmeans()` is efficient in that it doesn’t reduce until the closest k to the information threshold is found.

Usage

```r
reduce_kmeans(.partition_step, search = c("binary", "linear"), n_hits = 4)
```
reduce_scaled_mean

Arguments
- `.partition_step`  
  a partition_step object
- `search`  
  The search method. Binary search is generally more efficient but linear search can be faster in very low dimensions.
- `n_hits`  
  In linear search method, the number of iterations that should be under the threshold before reducing; useful for preventing false positives.

Value
- a partition_step object

See Also
Other reducers: `as_reducer()`, `reduce_first_component()`, `reduce_scaled_mean()`

reduce_scaled_mean  Reduce selected variables to scaled means

Description
Reducers are functions that tell the partition algorithm how to reduce the data. `as_reducer()` is a helper function to create new reducers to be used in partitioners. partitioners can be created with `as_partitioner()`.

reduce_scaled_mean() returns the scaled row means of the target variables to reduce.

Usage
reduce_scaled_mean(.partition_step)

Arguments
- `.partition_step`  
  a partition_step object

Value
- a partition_step object

See Also
Other reducers: `as_reducer()`, `reduce_first_component()`, `reduce_kmeans()`
replace_partitioner  Replace the director, metric, or reducer for a partitioner

Description
Replace the director, metric, or reducer for a partitioner

Usage
replace_partitioner(partitioner, direct = NULL, measure = NULL, reduce = NULL)

Arguments
- partitioner: a partitioner
- direct: a function that directs, possibly created by as_director()
- measure: a function that measures, possibly created by as_measure()
- reduce: a function that reduces, possibly created by as_reducer()

Value
a partitioner

See Also
Other partitioners: as_partitioner(), part_icc(), part_kmeans(), part_minr2(), part_pcl(), part_stdmi()

Examples
replace_partitioner(
  part_icc,
  reduce = as_reducer(rowMeans)
)
scaled_mean

Average and scale rows in a data.frame

Description

scaled_mean() calculates scaled row means for a dataframe.

Usage

scaled_mean(.x, method = c("r", "c"))

Arguments

.x a data.frame
method The method source: both the pure R and C++ versions are efficient

Value

a numeric vector

Examples

library(dplyr)
iris %>%
  select_if(is.numeric) %>%
  scaled_mean()

simulate_block_data

Simulate correlated blocks of variables

Description

simulate_block_data() creates a dataset of blocks of data where variables within each block are correlated. The correlation for each pair of variables is sampled uniformly from lower_corr to upper_corr, and the values of each are sampled using MASS::mvrnorm().

Usage

simulate_block_data(
  block_sizes,
  lower_corr,
  upper_corr,
  n,
  block_name = "block",
  sep = "_",
  var_name = "x"
)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>block_sizes</td>
<td>a vector of block sizes. The size of each block is the number of variables within it.</td>
</tr>
<tr>
<td>lower_corr</td>
<td>the lower bound of the correlation within each block</td>
</tr>
<tr>
<td>upper_corr</td>
<td>the upper bound of the correlation within each block</td>
</tr>
<tr>
<td>n</td>
<td>the number of observations or rows</td>
</tr>
<tr>
<td>block_name</td>
<td>description prepended to the variable to indicate the block it belongs to</td>
</tr>
<tr>
<td>sep</td>
<td>a character, what to separate the variable names with</td>
</tr>
<tr>
<td>var_name</td>
<td>the name of the variable within the block</td>
</tr>
</tbody>
</table>

Value

a tibble with \( \text{sum(block\_sizes)} \) columns and \( n \) rows.

Examples

```r
# create a 100 x 15 data set with 3 blocks
simulate_block_data(
  block_sizes = rep(5, 3),
  lower_corr = .4,
  upper_corr = .6,
  n = 100
)
```

Description

test_permutation() permutes data and partitions the results to generate a distribution of null statistics for observed information, number of clusters, and number of observed variables reduced to clusters. The result is a tibble with a summary of the observed data results and the averages of the permuted results. The partitions and and permutations are also available in list-cols. test_permutation() tests across a range of target information values, as specified in the information argument.

Usage

```r
test_permutation(
  .data,
  information = seq(0.1, 0.6, by = 0.1),
  partitioner = part_icc(),
  ...
)
```
**Arguments**

- `.data` a data set to partition
- `information` a vector of minimum information to fit in `partition()`
- `partitioner` the partitioner to use. The default is `part_icc()`.
- `...` arguments passed to `partition()`
- `nperm` Number of permuted data sets to test. Default is 100.

**Value**

a tibble with summaries on observed and permuted data (the means of the permuted summaries), as well as list-cols containing them
Index

* datasets
  baxter_data, 8
  baxter_data_dictionary (baxter_data), 8
  baxter_family (baxter_data), 8
  baxter_genus (baxter_data), 8
  baxter_otu (baxter_data), 8
  corr, 9

* directors
  as_director, 3
  direct_distance, 9
  direct_k_cluster, 10

* metrics
  as_measure, 4
  measure_icc, 16
  measure_min_icc, 17
  measure_min_r2, 17
  measure_std_mutualinfo, 18
  measure_variance_explained, 19
  direct_distance, 3, 9, 11
  direct_distance_pearson (as_measure), 9
  direct_distance_spearman (as_measure), 9
  direct_k_cluster, 3, 10, 10

* partitioners
  as_partitioner, 5
  part_icc, 23
  part_kmeans, 24
  part_minr2, 25
  part_pc1, 26
  part_stdmi, 27
  replace_partitioner, 34
  filter_reduced, 11
  fitted.partition (partition_scores), 22
  icc, 12
  is_partition, 13
  is_partition_step, 14
  is_partitioner, 13
  kmeans(), 11, 24
  map_cluster (reduce_cluster), 31
  map_partition, 15
  map_partition(), 28, 29
  mapping_groups (mapping_key), 14
  mapping_key, 14
  MASS::mvrnorm(), 35
  measure_icc, 4, 16–17–19
  measure_min_icc, 4, 16, 17, 18, 19
  measure_min_r2, 4, 16, 17, 18, 19
  measure_std_mutualinfo, 4, 16–18, 18, 19
  measure_variance_explained, 4, 16–18, 19
  mutual_information, 19
  mutual_information(), 18

* reducers
  as_reducer, 7
  reduce_first_component, 32
  reduce_kmeans, 32
  reduce_scaled_mean, 33
  as_director, 3, 10, 11
  as_director(), 3, 5, 9, 10, 34
  as_measure, 4, 16–19
  as_measure(), 4, 5, 16–19, 34
  as_partition_step, 6
  as_partitioner, 5, 23–27, 34
  as_partitioner(), 3–7, 9, 10, 16–21, 23–27, 29, 32, 33
  as_reducer, 7, 32, 33
  as_reducer(), 5, 7, 31–34

  baxter_clinical (baxter_data), 8
  baxter_data, 8
  baxter_data_dictionary (baxter_data), 8
  baxter_family (baxter_data), 8
  baxter_genus (baxter_data), 8
  baxter_otu (baxter_data), 8
  corr, 9
  direct_distance, 3, 9, 11
  direct_distance_pearson (as_measure), 9
  direct_distance_spearman (as_measure), 9
  direct_k_cluster, 3, 10, 10
  filter_reduced, 11
  fitted.partition (partition_scores), 22
  icc, 12
  is_partition, 13
  is_partition_step, 14
  is_partitioner, 13
  kmeans(), 11, 24
  map_cluster (reduce_cluster), 31
  map_partition, 15
  map_partition(), 28, 29
  mapping_groups (mapping_key), 14
  mapping_key, 14
  MASS::mvrnorm(), 35
  measure_icc, 4, 16–17–19
  measure_min_icc, 4, 16, 17, 18, 19
  measure_min_r2, 4, 16, 17, 18, 19
  measure_std_mutualinfo, 4, 16–18, 18, 19
  measure_variance_explained, 4, 16–18, 19
  mutual_information, 19
  mutual_information(), 18
  part_icc, 5, 23, 24–27, 34
  part_icc(), 15, 21, 37
  part_kmeans, 5, 23, 24, 25–27, 34
INDEX

part_kmeans(), 21
part_minr2, 5, 23, 24, 25, 26, 27, 34
part_minr2(), 21
part_pcl, 5, 23–25, 26, 27, 34
part_pcl(), 21
part_stdmi, 5, 23–26, 27, 34
part_stdmi(), 21
partition, 20
partition(), 5, 15, 23–27, 29, 37
partition_scores, 22
permute_df, 28
plot_area_clusters, 28
plot_information(plot_area_clusters), 28
plot_ncluster(plot_area_clusters), 28
plot_permutation, 30
plot_stacked_area_clusters
    (plot_area_clusters), 28
reduce_cluster, 31
reduce_first_component, 7, 32, 33
reduce_kmeans, 7, 32, 32, 33
reduce_scaled_mean, 7, 32, 33, 33
replace_partitioner, 5, 23–27, 34
replace_partitioner(), 21
scaled_mean, 35
simulate_block_data, 35
test_permutation, 36
test_permutation(), 28–30
unnest_mappings(mapping_key), 14
unnest_reduced(filter_reduced), 11