

Package ‘partition’

May 17, 2019

Type Package

Title Agglomerative Partitioning Framework for Dimension Reduction

Version 0.1.0

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.

Depends R (>= 3.3.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp, rlang, purrr, tibble, dplyr (>= 0.8.0), tidyr, ggplot2 (>= 3.0.0), stringr, magrittr, crayon, pillar, infotheo, MASS, forcats

Suggests covr, spelling, testthat, knitr, rmarkdown

RoxygenNote 6.1.1

URL <https://uscbiostats.github.io/partition/>,
<https://github.com/USCbiostats/partition>

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

Language en-US

VignetteBuilder knitr

NeedsCompilation yes

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

Repository CRAN

Date/Publication 2019-05-17 07:00:04 UTC

R topics documented:

as_director	3
as_measure	4
as_partitioner	5
as_partition_step	6
as_reducer	7
corr	8
direct_distance	8
direct_k_cluster	9
filter_reduced	10
icc	11
increase_hits	11
is_partition	12
is_partitioner	12
is_partition_step	13
mapping_key	13
map_partition	14
measure_icc	15
measure_min_icc	15
measure_min_r2	16
measure_std_mutualinfo	17
measure_variance_explained	17
mutual_information	18
partition	19
partition_scores	20
part_icc	21
part_kmeans	22
part_minr2	23
part_pc1	24
part_stdmi	25
permute_df	26
plot_area_clusters	26
plot_permutation	28
reduce_cluster	29
reduce_first_component	30
reduce_kmeans	30
reduce_scaled_mean	31
replace_partitioner	32
scaled_mean	33
simulate_block_data	33
test_permutation	34

`as_director`*Create a custom 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

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

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. `reduce`. `as_measure()` is a helper function to create new metrics to be used in partitioners. partitioners can be created with `as_partitioner()`.

Usage

```
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](#)

Other metrics: [measure_icc](#), [measure_min_icc](#), [measure_min_r2](#), [measure_std_mutualinfo](#), [measure_variance_explained](#)

Examples

```
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_pc1](#), [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)
```

as_reducer	<i>Create a custom reducer</i>
------------	--------------------------------

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

<code>.f</code>	a function that returns either a numeric vector or a data.frame
<code>...</code>	Extra arguments passed to <code>.f</code> .
<code>returns_vector</code>	logical. Does <code>.f</code> return a vector? TRUE by default. If FALSE, assumes that <code>.f</code> returns a data.frame.
<code>first_match</code>	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](#)

Other reducers: [reduce_first_component](#), [reduce_kmeans](#), [reduce_scaled_mean](#)

Examples

```
reduce_row_means <- as_reducer(rowMeans)
reduce_row_means
```

corr	<i>Efficiently fit correlation coefficient for matrix or two vectors</i>
------	--------------------------------------------------------------------------

Description

Efficiently fit correlation coefficient for matrix or two vectors

Usage

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

Arguments

x	a matrix or vector
y	a vector. Optional.
spearman	Logical. Use Spearman's correlation?

Value

a numeric vector, the correlation coefficient

Examples

```
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	<i>Target based on minimum distance matrix</i>
-----------------	------------------------------------------------

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.

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	<i>Target based on K-means clustering</i>
------------------	-------------------------------------------

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)
```

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	<i>Filter the reduced mappings</i>
----------------	------------------------------------

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

```
filter_reduced(.partition)
```

```
unnest_reduced(.partition)
```

Arguments

.partition a partition object

Value

a tibble with mapping key

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)
```

```
# 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

```
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

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

increase_hits *Count and retrieve the number of metrics below threshold*

Description

Count and retrieve the number of metrics below threshold

Usage

```
increase_hits(.partition_step)

get_hits(.partition_step)
```

Arguments

.partition_step	a partition_step object
-----------------	-------------------------

is_partition	<i>Is this object a partition?</i>
--------------	------------------------------------

Description

Is this object a partition?

Usage

```
is_partition(x)
```

Arguments

x an object to be tested

Value

logical: TRUE or FALSE

is_partitioner	<i>Is this object a partitioner?</i>
----------------	--------------------------------------

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	<i>Is this object a partition_step?</i>
-------------------	-----------------------------------------

Description

Is this object a partition_step?

Usage

```
is_partition_step(x)
```

Arguments

x an object to be tested

Value

logical: TRUE or FALSE

mapping_key	<i>Return partition mapping key</i>
-------------	-------------------------------------

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() unnests 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

```
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

```

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

```

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

```

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	<i>Measure the information loss of reduction using intraclass correlation coefficient</i>
-------------	-------------------------------------------------------------------------------------------

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_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	<i>Measure the information loss of reduction using the minimum intraclass correlation coefficient</i>
-----------------	-------------------------------------------------------------------------------------------------------

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)
```

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

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

partition	<i>Agglomerative partitioning</i>
-----------	-----------------------------------

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

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

Arguments

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

See Also

[part_icc\(\)](#), [part_kmeans\(\)](#), [part_minr2\(\)](#), [part_pc1\(\)](#), [part_stdmi\(\)](#), [as_partitioner\(\)](#), [replace_partitioner\(\)](#)

Examples

```
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	<i>Return the reduced data from a partition</i>
------------------	-------------------------------------------------

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 <code>fitted()</code>)

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

```
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

```
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())
```

part_kmeans

Partitioner: K-means, 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_kmeans()` uses the following direct-measure-reduce approach:

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

Usage

```
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 <code>armadillo</code> . The rest are options in kmeans() . In general, <code>armadillo</code> 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

```
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

```
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

```
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

```
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`

Examples

```
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

```
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](#)

Examples

```
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	<i>Permute a data set</i>
------------	---------------------------

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

```
permute_df(.data)
```

Arguments

.data	a data.frame
-------	--------------

Value

a permuted data.frame

Examples

```
permute_df(iris)
```

plot_area_clusters	<i>Plot partitions</i>
--------------------	------------------------

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

<code>.data</code>	a data.frame to partition
<code>partitioner</code>	a partitioner. See the <code>part_*()</code> functions and <code>as_partitioner()</code> .
<code>information</code>	a vector of minimum information to fit in <code>partition()</code>
<code>...</code>	arguments passed to <code>partition()</code>
<code>obs_color</code>	the color of the observed partition
<code>perm_color</code>	the color of the permuted partition
<code>stack_colors</code>	the colors of the cluster sizes
<code>.partition</code>	either a partition or a tibble, the result of <code>map_partition()</code> or <code>test_permutation()</code>
<code>show_n</code>	the number of reduced variables to plot
<code>fill</code>	the color of the fill for geom
<code>color</code>	the color of the geom
<code>labeller</code>	the facet label
<code>geom</code>	the geom to use. The default is <code>geom_density</code> .

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()

```

plot_permutation	<i>Plot permutation tests</i>
------------------	-------------------------------

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 <code>test_permutation()</code>
.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 <code>geom_density</code> .

Value

a ggplot

Examples

```
set.seed(123)  
df <- simulate_block_data(c(3, 4, 5), lower_corr = .4, upper_corr = .6, n = 100)  
  
test_permutation(df, partitioner = part_pc1(), nperm = 5) %>%  
  plot_permutation("nclusters")
```

reduce_cluster	<i>Reduce a target</i>
----------------	------------------------

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

<code>.partition_step</code>	a <code>partition_step</code> object
<code>.f</code>	a function to reduce the data to either a vector or a <code>data.frame</code>
<code>first_match</code>	logical. Should the partition algorithm stop when it finds a reduction that is equal to the threshold? Default is <code>TRUE</code> for reducers that return a <code>data.frame</code> and <code>FALSE</code> for reducers that return a vector
<code>rewind</code>	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

```
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

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

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	<i>Reduce selected variables to scaled means</i>
--------------------	--------------------------------------------------

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_pc1](#), [part_stdmi](#)

Examples

```
replace_partitioner(  
  part_icc,  
  reduce = as_reducer(rowMeans)  
)
```

scaled_mean	<i>Average and scale rows in a data.frame</i>
-------------	-----------------------------------------------

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	<i>Simulate correlated blocks of variables</i>
---------------------	------------------------------------------------

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

block_sizes	a vector of block sizes. The size of each block is the number of variables within it.
lower_corr	the lower bound of the correlation within each block
upper_corr	the upper bound of the correlation within each block
n	the number of observations or rows
block_name	description prepended to the variable to indicate the block it belongs to
sep	a character, what to separate the variable names with
var_name	the name of the variable within the block

Value

a tibble with `sum(block_sizes)` columns and `n` rows.

Examples

```
# 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
)
```

test_permutation	<i>Permute partitions</i>
------------------	---------------------------

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 permutations are also available in `list-cols`. `test_permutation()` tests across a range of target information values, as specified in the `information` argument.

Usage

```
test_permutation(.data, information = seq(0.1, 0.6, by = 0.1),
  partitioner = part_icc(), ..., nperm = 100)
```

Arguments

<code>.data</code>	a data set to partition
<code>information</code>	a vector of minimum information to fit in <code>partition()</code>
<code>partitioner</code>	the partitioner to use. The default is <code>part_icc()</code> .
<code>...</code>	arguments passed to <code>partition()</code>
<code>nperm</code>	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

Examples

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

test_permutation(df, partitioner = part_pc1(), nperm = 10)
```

Index

`as_director`, 3, 9, 10
`as_director()`, 3, 5, 8, 9, 32
`as_measure`, 4, 15–18
`as_measure()`, 4, 5, 15–17, 32
`as_partition_step`, 6
`as_partitioner`, 5, 22–25, 32
`as_partitioner()`, 3–9, 15–17, 19–25, 27, 30, 31
`as_reducer`, 7, 30, 31
`as_reducer()`, 5, 7, 29–32

`corr`, 8

`direct_distance`, 3, 8, 10
`direct_distance_pearson`
 (`direct_distance`), 8
`direct_distance_spearman`
 (`direct_distance`), 8
`direct_k_cluster`, 3, 9, 9

`filter_reduced`, 10
`fitted.partition(partition_scores)`, 20

`get_hits(increase_hits)`, 11

`icc`, 11
`increase_hits`, 11
`is_partition`, 12
`is_partition_step`, 13
`is_partitioner`, 12

`kmeans()`, 9, 22

`map_cluster(reduce_cluster)`, 29
`map_partition`, 14
`map_partition()`, 26, 27
`mapping_groups(mapping_key)`, 13
`mapping_key`, 13
`MASS::mvrnorm()`, 33
`measure_icc`, 4, 15, 16–18
`measure_min_icc`, 4, 15, 15, 16–18

`measure_min_r2`, 4, 15, 16, 16, 17, 18
`measure_std_mutualinfo`, 4, 15, 16, 17, 18
`measure_variance_explained`, 4, 15–17, 17
`mutual_information`, 18
`mutual_information()`, 17

`part_icc`, 5, 21, 23–25, 32
`part_icc()`, 14, 19, 20, 35
`part_kmeans`, 5, 22, 22, 23–25, 32
`part_kmeans()`, 20
`part_minr2`, 5, 22, 23, 23, 24, 25, 32
`part_minr2()`, 20
`part_pc1`, 5, 22, 23, 24, 25, 32
`part_pc1()`, 20
`part_stdmi`, 5, 22–24, 25, 32
`part_stdmi()`, 20
`partition`, 19
`partition()`, 5, 14, 21–25, 27, 35
`partition_scores`, 20
`permute_df`, 26
`plot_area_clusters`, 26
`plot_information(plot_area_clusters)`, 26
`plot_ncluster(plot_area_clusters)`, 26
`plot_permutation`, 28
`plot_stacked_area_clusters`
 (`plot_area_clusters`), 26

`reduce_cluster`, 29
`reduce_first_component`, 7, 30, 31
`reduce_kmeans`, 7, 30, 30, 31
`reduce_scaled_mean`, 7, 30, 31, 31
`replace_partitioner`, 5, 22–25, 32
`replace_partitioner()`, 19, 20

`scaled_mean`, 33
`simulate_block_data`, 33

`test_permutation`, 34
`test_permutation()`, 26–28

`unnest_mappings(mapping_key)`, [13](#)
`unnest_reduced(filter_reduced)`, [10](#)