**create_folds**

### Description

This function provides a list of row indices per fold of k-fold cross-validation (basic, stratified, grouped, or blocked). Repeated fold creation is supported as well.

### Usage

```r
create_folds(
  y,
  k = 5,
  type = c("stratified", "basic", "grouped", "blocked"),
  n_bins = 10,
  m_rep = 1,
  use_names = TRUE,
  invert = FALSE,
  shuffle = FALSE,
  seed = NULL
)
```

### Arguments

- **y**: Either the variable used for "stratification" or "grouped" splits. For other types of splits, any vector of the same length as the data intended to split.
- **k**: Number of folds.
- **type**: Split type. One of "stratified", "basic", "grouped", "blocked". The default is "stratified".
- **n_bins**: Approximate numbers of bins for numeric `y` and `type = "stratified"`.
- **m_rep**: How many times should the data be split into k folds? Default is 1, i.e. no repetitions.
- **use_names**: Should folds be named? Default is `TRUE`.
- **invert**: Set to `TRUE` if the row numbers not in the fold are to be returned. Default is `FALSE`.
- **shuffle**: Should row indices be randomly shuffled within folds? Default is `FALSE`.
- **seed**: Integer random seed.
Details

By default, the function uses stratified splitting. This will balance the folds regarding the distribution of the input vector \( y \). Numeric input is first binned into \( n_{\text{bins}} \) quantile groups. If \( \text{type} = \) "grouped", groups specified by \( y \) are kept together when splitting. This is relevant for clustered or panel data. In contrast to basic splitting, \( \text{type} = \) "blocked" does not sample indices at random, but rather keeps them in sequential groups.

Value

A list with row indices per fold.

Examples

```r
y <- rep(c(letters[1:4]), each = 5)
create_folds(y)
create_folds(y, k = 2)
create_folds(y, k = 2, m_rep = 2)
create_folds(y, k = 3, type = "blocked")
```

create_timefolds

Creates Folds for Time Series Data

Description

This function provides a list with in- and out-of-sample indices per fold used for time series k-fold cross-validation, see Details.

Usage

```r
create_timefolds(y, k = 5, use_names = TRUE, type = c("extending", "moving"))
```

Arguments

- `y`: Any vector of the same length as the data intended to split.
- `k`: Number of folds.
- `use_names`: Should folds be named? Default is `TRUE`.
- `type`: Should insample data be "extending" over the folds (default) or consist of one single fold ("moving")?

Details

The data is first partitioned into \( k+1 \) sequential blocks \( B_1 \) to \( B_{k+1} \). Each fold consists of two index vectors: one with in-sample row numbers, the other with out-of-sample row numbers. The first fold uses \( B_1 \) as in-sample and \( B_2 \) as out-of-sample data. The second one uses either \( B_2 \) (if \( \text{type} = \) "extending") or \( B_1, B_2 \) (if \( \text{type} = \) "moving") as in-sample and \( B_3 \) as out-of-sample data etc. until the kth fold with \( B_1, ..., B_k \) ("extending") resp. \( B_k \) ("moving") as in-sample and \( B_{k+1} \) as out-of-sample data. This makes sure that out-of-sample data always follows in-sample data.
partition

Value

A nested list with in-sample ("insample") and out-of-sample ("outsample") indices per fold.

Examples

```r
y <- runif(100)
create_timefolds(y)
create_timefolds(y, use_names = FALSE)
create_timefolds(y, use_names = FALSE, type = "moving")
```

Description

This function provides row indices for data splitting, e.g. into training, validation, and test. Different types of split strategies are supported ("basic", "stratified" (the default), "grouped", or "blocked"), see Details. The partition indices are either returned as a list with one element per partition (the default) or as vector of partition ids.

Usage

```r
partition(
  y,
  p,
  type = c("stratified", "basic", "grouped", "blocked"),
  n_bins = 10,
  split_into_list = TRUE,
  use_names = TRUE,
  shuffle = FALSE,
  seed = NULL
)
```

Arguments

- **y**: Either the variable used for "stratification" or "grouped" splits. For other types of splits, any vector of the same length as the data intended to split.
- **p**: A vector with split probabilities per partition, e.g. `c(train = 0.7, valid = 0.3)`. Names are passed to the output.
- **type**: Split type. One of "stratified", "basic", "grouped", "blocked". The default is "stratified".
- **n_bins**: Approximate numbers of bins for numeric `y` and type = "stratified".
- **split_into_list**: Should the resulting partition vector be split into a list or not? Default is `TRUE`.
- **use_names**: Should names of `p` be used as partition names? Default is `TRUE`.
shuffle Should row indices be randomly shuffled within partition? Default is FALSE. Shuffling is only possible when split_into_list = TRUE.

seed Integer random seed.

Details

By default, the function uses stratified splitting. This will balance the partitions as good as possible regarding the distribution of the input vector y. Numeric input is first binned into n_bins quantile groups. If type = "grouped", groups specified by y are kept together when splitting. This is relevant for clustered or panel data. In contrast to basic splitting, type = "blocked" does not sample indices at random, but rather keeps them in groups: e.g. the first 80% of observations form a training set and the remaining 20% are used for testing.

Value

A list with row indices per partition (if split_into_list = TRUE) or a vector of partition ids.

Examples

```r
y <- rep(c(letters[1:4]), each = 5)
partition(y, p = c(0.7, 0.3), seed = 1)
partition(y, p = c(0.7, 0.3), split_into_list = FALSE, seed = 1)
p <- c(train = 0.8, valid = 0.1, test = 0.1)
partition(y, p, seed = 1)
partition(y, p, split_into_list = FALSE, seed = 1)
partition(y, p, split_into_list = FALSE, use_names = FALSE, seed = 1)
partition(y, p = c(0.7, 0.3), type = "grouped")
partition(y, p = c(0.7, 0.3), type = "blocked")
```
Index

create_folds, 2
create_timefolds, 3

partition, 4